Aries Cloud Agent Python Documentation

See Contributors list on GitHub

Apr 06, 2023

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

ONE

ARIES_CLOUDAGENT

1.1 aries_cloudagent package

Aries Cloud Agent.

1.1.1 Subpackages

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.

abstract async start() → None

Start the webserver.

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

abstract async stop() → None

Stop the webserver.

aries_cloudagent.admin.error module

Admin error classes.

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

Bases: *aries_cloudagent.core.error.BaseError*

Base class for Admin-related errors.

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

Bases: *aries_cloudagent.admin.error.AdminError*

Admin server setup or configuration error.

aries_cloudagent.admin.request_context module

Admin request context class.

A request context provided by the admin server to admin route handlers.

class aries_cloudagent.admin.request_context.**AdminRequestContext**(*profile: aries_cloudagent.core.profile.Profile, *, context: Optional[aries_cloudagent.config.injection_context.I = None, settings: Optional[Mapping[str, object]] = None*)

Bases: object

Context established by the Conductor and passed into message handlers.

inject(*base_cls: Type[aries_cloudagent.config.base.InjectType], settings: Optional[Mapping[str, object]] = None*) → aries_cloudagent.config.base.InjectType

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

inject_or(*base_cls: Type[aries_cloudagent.config.base.InjectType], settings: Optional[Mapping[str, object]] = None, default: Optional[aries_cloudagent.config.base.InjectType] = None*) → Optional[aries_cloudagent.config.base.InjectType]

Get the provided instance of a given class identifier or default if not found.

Parameters

- **base_cls** – The base class to retrieve an instance of
- **settings** – An optional dict providing configuration to the provider
- **default** – default return value if no instance is found

Returns An instance of the base class, or None

property injector: *aries_cloudagent.config.injector.Injector*

Accessor for the associated *Injector* instance.

property profile: *aries_cloudagent.core.profile.Profile*

Accessor for the associated *Profile* instance.

session() → *aries_cloudagent.core.profile.ProfileSession*

Start a new interactive session with no transaction support requested.

property settings: *aries_cloudagent.config.settings.Settings*

Accessor for the context settings.

classmethod test_context(*session_inject: Optional[dict] = None, profile: Optional[aries_cloudagent.core.profile.Profile] = None*) → *aries_cloudagent.admin.request_context.AdminRequestContext*

Quickly set up a new admin request context for tests.

transaction() → *aries_cloudagent.core.profile.ProfileSession*

Start a new interactive session with commit and rollback support.

If the current backend does not support transactions, then commit and rollback operations of the session will not have any effect.

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

aries_cloudagent.admin.server module

aries_cloudagent.askar package

Subpackages

aries_cloudagent.askar.didcomm package

Submodules

aries_cloudagent.askar.didcomm.v1 module

DIDComm v1 envelope handling via Askar backend.

aries_cloudagent.askar.didcomm.v1.**pack_message**(*to_verkeys: Sequence[str]*, *from_key: Optional[aries_askar.Key]*, *message: bytes*) → bytes
Encode a message using the DIDComm v1 'pack' algorithm.

async aries_cloudagent.askar.didcomm.v1.**unpack_message**(*session: aries_askar.Session*, *enc_message: bytes*) → Tuple[str, str, str]
Decode a message using the DIDComm v1 'unpack' algorithm.

aries_cloudagent.askar.didcomm.v2 module

DIDComm v2 envelope handling via Askar backend.

exception aries_cloudagent.askar.didcomm.v2.**DidcommEnvelopeError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.wallet.error.WalletError*

A base error class for DIDComm envelope wrapping and unwrapping operations.

aries_cloudagent.askar.didcomm.v2.**ecdh_1pu_decrypt**(*wrapper:* aries_cloudagent.utils.jwe.JweEnvelope, *recip_kid: str*, *recip_key: aries_askar.Key*, *sender_key: aries_askar.Key*) → Tuple[str, str, str]
Decode a message with DIDComm v2 authenticated encryption.

aries_cloudagent.askar.didcomm.v2.**ecdh_1pu_encrypt**(*to_verkeys: Mapping[str, aries_askar.Key]*, *sender_kid: str*, *sender_key: aries_askar.Key*, *message: bytes*) → bytes
Encode a message using DIDComm v2 authenticated encryption.

aries_cloudagent.askar.didcomm.v2.**ecdh_es_decrypt**(*wrapper:* aries_cloudagent.utils.jwe.JweEnvelope, *recip_kid: str*, *recip_key: aries_askar.Key*) → bytes
Decode a message with DIDComm v2 anonymous encryption.

aries_cloudagent.askar.didcomm.v2.**ecdh_es_encrypt**(*to_verkeys: Mapping[str, aries_askar.Key]*, *message: bytes*) → bytes

Encode a message using DIDComm v2 anonymous encryption.

async aries_cloudagent.askar.didcomm.v2.**unpack_message**(*session: aries_askar.Session*, *enc_message: Union[bytes, str]*) → Tuple[str, str, str]

Decode a message using DIDComm v2 encryption.

Submodules

aries_cloudagent.askar.profile module

aries_cloudagent.askar.store module

Aries-Askar backend store configuration.

class aries_cloudagent.askar.store.**AskarOpenStore**(*config:* aries_cloudagent.askar.store.AskarStoreConfig, *created*, *store: aries_askar.Store*)

Bases: object

Handle and metadata for an opened Askar store.

async close()

Close previously-opened store, removing it if so configured.

property name: str

Accessor for the store name.

class aries_cloudagent.askar.store.**AskarStoreConfig**(*config: Optional[dict] = None*)

Bases: object

A helper class for handling Askar store configuration.

DEFAULT_KEY = ''

DEFAULT_KEY_DERIVATION = 'kdf:argon2i:mod'

DEFAULT_STORAGE_TYPE = None

KEY_DERIVATION_ARGON2I_INT = 'kdf:argon2i:int'

KEY_DERIVATION_ARGON2I_MOD = 'kdf:argon2i:mod'

KEY_DERIVATION_RAW = 'RAW'

get_uri(*create: bool = False*) → str

Accessor for the storage URI.

async open_store(*provision: bool = False*) → *aries_cloudagent.askar.store.AskarOpenStore*

Open a store, removing and/or creating it if so configured.

Raises

- **ProfileNotFoundError** – If the store is not found
- **ProfileError** – If there is another aries_askar error

async remove_store()

Remove an existing store.

Raises

- **ProfileNotFoundError** – If the wallet could not be found
- **ProfileError** – If there was another aries_askar error

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.

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

Parameters **key** – the key to remove

abstract async flush()
Remove all items from the cache.

abstract async 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.

abstract async set(*keys: Union[str, Sequence[str]]*, *value: Any*, *ttl: Optional[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: Optional[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.

property done: bool
Accessor for the done state.

property future: _asyncio.Future
Fetch the result in the form of an awaitable future.

property parent: *aries_cloudagent.cache.base.CacheKeyLock*
Accessor for the parent key lock, if any.

release()
Release the cache lock.

property result: Any
Fetch the current result, if any.

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

aries_cloudagent.cache.in_memory module

Basic in-memory cache implementation.

class aries_cloudagent.cache.in_memory.**InMemoryCache**
Bases: *aries_cloudagent.cache.base.BaseCache*

Basic in-memory cache class.

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

Parameters **key** – the key to remove

async flush()
Remove all items from the cache.

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

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

Returns The record found or *None*

async set(*keys: Union[str, Sequence[str]]*, *value: Any*, *ttl: Optional[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

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: Optional[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: Optional[Sequence[str]] = None*)
Execute the help command.

aries_cloudagent.commands.help.**main**()
Execute the main line.

aries_cloudagent.commands.provision module

aries_cloudagent.commands.start module

aries_cloudagent.commands.upgrade module

aries_cloudagent.config package

Submodules

aries_cloudagent.config.argparse module

aries_cloudagent.config.banner module

Module to contain logic to generate the banner for ACA-py.

class aries_cloudagent.config.banner.**Banner**(*border: str*, *length: int*)
Bases: `object`

Management class to generate a banner for ACA-py.

lr_pad(*content: str*)
Pad string content with defined border character.

Parameters **content** – String content to pad

print_border()
Print a full line using the border character.

print_list(*items*)
Print a list of items, prepending a dash to each item.

print_spacer()
Print an empty line with the border character only.

print_subtitle(*title*)
Print a subtitle for a section.

print_title(*title*)
Print the main title element.

print_version(*version*)
Print the current `version`.

aries_cloudagent.config.base module

Configuration base classes.

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

Base injector class.

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

abstract inject(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, Any]] = None*) → aries_cloudagent.config.base.InjectType
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

abstract inject_or(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, Any]] = None*, *default: Optional[aries_cloudagent.config.base.InjectType] = None*) → Optional[aries_cloudagent.config.base.InjectType]
Get the provided instance of a given class identifier or default if not found.

Parameters

- **base_cls** – The base class to retrieve an instance of
- **settings** – An optional dict providing configuration to the provider
- **default** – default return value if no instance is found

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**(**args*, ***kwds*)

Bases: Mapping[str, Any]

Base settings class.

abstract copy() → *aries_cloudagent.config.base.BaseSettings*

Produce a copy of the settings instance.

abstract extend(*other: Mapping[str, Any]*) → *aries_cloudagent.config.base.BaseSettings*

Merge another mapping to produce a new settings instance.

get_bool(**var_names*, *default: Optional[bool] = None*) → Optional[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: Optional[int] = None*) → Optional[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: Optional[str] = None*) → Optional[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

abstract get_value(**var_names*, *default: Optional[Any] = None*) → Any

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: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

A base exception for all configuration errors.

exception aries_cloudagent.config.base.**InjectionError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.config.base.ConfigError*

The base exception raised by Injector and Provider implementations.

exception aries_cloudagent.config.base.**SettingsError**(**args*, *error_code: Optional[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: Optional[Mapping[str, Any]] = None*)

Bases: abc.ABC

Base injection context builder class.

abstract async build_context() → *aries_cloudagent.config.injection_context.InjectionContext*
Build the base 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: Optional[Mapping[str, Any]] = None*)

Bases: *aries_cloudagent.config.base_context.ContextBuilder*

Default context builder.

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

async build_context() → *aries_cloudagent.config.injection_context.InjectionContext*
Build the base injection context; set DIDComm prefix to emit.

async 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: Optional[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: Optional[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[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*) → aries_cloudagent.config.base.InjectType
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

inject_or(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*, *default: Optional[aries_cloudagent.config.base.InjectType] = None*) → Optional[aries_cloudagent.config.base.InjectType]
Get the provided instance of a given class identifier or default if not found.

Parameters

- **base_cls** – The base class to retrieve an instance of
- **settings** – An optional dict providing configuration to the provider
- **default** – default return value if no instance is found

Returns An instance of the base class, or None

property injector: *aries_cloudagent.config.injector.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

property scope_name: str
Accessor for the current scope name.

property settings: *aries_cloudagent.config.settings.Settings*
Accessor for scope-specific settings.

start_scope(*scope_name: str*, *settings: Optional[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: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.config.base.InjectionError*

Base class for issues in the injection context.

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

property injector
Alias for field number 1

property name
Alias for field number 0

aries_cloudagent.config.injector module

Standard Injector implementation.

class aries_cloudagent.config.injector.**Injector**(*settings: Optional[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[aries_cloudagent.config.base.InjectType]*, *instance: aries_cloudagent.config.base.InjectType*)
Add a static instance as a class binding.

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

clear_binding(*base_cls: Type[aries_cloudagent.config.base.InjectType]*)
Remove a previously-added binding.

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

get_provider(*base_cls: Type[aries_cloudagent.config.base.InjectType]*)
Find the provider associated with a class binding.

inject(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*) → aries_cloudagent.config.base.InjectType
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

inject_or(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*, *default: Optional[aries_cloudagent.config.base.InjectType] = None*) → Optional[aries_cloudagent.config.base.InjectType]
Get the provided instance of a given class identifier or default if not found.

Parameters

- **base_cls** – The base class to retrieve an instance of
- **settings** – An optional dict providing configuration to the provider
- **default** – default return value if no instance is found

Returns An instance of the base class, or None

property settings: *aries_cloudagent.config.settings.Settings*
Accessor for scope-specific settings.

aries_cloudagent.config.ledger module

Ledger configuration.

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

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

async aries_cloudagent.config.ledger.**get_genesis_transactions**(*settings:* aries_cloudagent.config.settings.Settings) → str
Fetch genesis transactions if necessary.

async aries_cloudagent.config.ledger.**ledger_config**(*profile:* aries_cloudagent.core.profile.Profile, *public_did:* *str*, *provision:* *bool* = *False*) → bool
Perform Indy ledger configuration.

async aries_cloudagent.config.ledger.**load_multiple_genesis_transactions_from_config**(*settings:* aries_cloudagent.config.s
Fetch genesis transactions for multiple ledger configuration.

async aries_cloudagent.config.ledger.**select_aml_tty**(*taa_info*, *provision:* *bool* = *False*) → Optional[str]
Select acceptance mechanism from AML.

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: Optional[str]* = *None*, *log_level: Optional[str]* = *None*, *log_file: Optional[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: Optional[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.plugin_settings module

Settings implementation for plugins.

class aries_cloudagent.config.plugin_settings.**PluginSettings**(*values: Optional[Mapping[str, Any]] = None*)

Bases: Mapping[str, Any]

Retrieve immutable settings for plugins.

Plugin settings should be retrieved by calling:

PluginSettings.for_plugin(settings, "my_plugin", {"default": "values"})

This will extract the PLUGIN_CONFIG_KEY in "settings" and return a new PluginSettings instance.

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

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

classmethod for_plugin(*settings:* aries_cloudagent.config.base.BaseSettings, *plugin: str*, *default: Optional[Mapping[str, Any]] = None*) → *aries_cloudagent.config.plugin_settings.PluginSettings*
Construct a PluginSettings object from another settings object.

PLUGIN_CONFIG_KEY is read from settings.

get_value(**var_names: str*, *default: Optional[Any] = None*)
Fetch a setting.

Parameters

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

aries_cloudagent.config.provider module

Service provider implementations.

class aries_cloudagent.config.provider.**CachedProvider**(*provider:* aries_cloudagent.config.base.BaseProvider, *unique_settings_keys: tuple = ()*)

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.

Instances are cached keyed on a SHA256 digest of the relevant subset of settings.

class aries_cloudagent.config.provider.**ClassProvider**(*instance_cls: Union[str, type]*, **ctor_args*, *init_method: Optional[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: Optional[Mapping[str, Any]] = None*)
Bases: *aries_cloudagent.config.base.BaseSettings*, MutableMapping[str, Any]

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, Any]*) → *aries_cloudagent.config.base.BaseSettings*
Merge another settings instance to produce a new instance.

for_plugin(*plugin: str*, *default: Optional[Mapping[str, Any]] = None*)
Retrieve settings for plugin.

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

update(*other: Mapping[str, Any]*)
Update the settings in place.

aries_cloudagent.config.util module

Entrypoint.

class aries_cloudagent.config.util.**BoundedInt**(*min: Optional[int] = None*, *max: Optional[int] = None*)
Bases: object

Argument value parser for a bounded integer.

class aries_cloudagent.config.util.**ByteSize**(*min: int = 0*, *max: Optional[int] = None*)
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.

async aries_cloudagent.config.wallet.**add_or_update_version_to_storage**(*session:*
aries_cloudagent.core.profile.ProfileSession

Add or update ACA-Py version StorageRecord.

async aries_cloudagent.config.wallet.**wallet_config**(*context:*
aries_cloudagent.config.injection_context.InjectionContext,
provision: bool = False) →
Tuple[*aries_cloudagent.core.profile.Profile*,
aries_cloudagent.wallet.did_info.DIDInfo]

Initialize the root profile.

aries_cloudagent.connections package

Subpackages

aries_cloudagent.connections.models package

Subpackages

aries_cloudagent.connections.models.diddoc package

DID Document model support.

class aries_cloudagent.connections.models.diddoc.**DIDDoc**(*did: Optional[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

property authnkey: dict

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

property did: str

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

property pubkey: dict

Accessor for public keys by identifier.

serialize() → dict

Dump current object to a JSON-compatible dictionary.

Returns dict representation of current DIDDoc

property service: dict

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

property authn_type
Alias for field number 1

property specifier
Alias for field number 2

property ver_type
Alias for field number 0

class aries_cloudagent.connections.models.diddoc.**PublicKey**(*did: str*, *ident: str*, *value: str*, *pk_type: Optional[*aries_cloudagent.connections.models.diddoc.public *= None*, *controller: Optional[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).

property authn: bool
Accessor for the authentication marker.

Returns: whether public key is marked as having DID authentication privilege

property controller: str
Accessor for the controller DID.

property did: str
Accessor for the DID.

property id: str
Accessor for the public key identifier.

to_dict() → dict
Return dict representation of public key to embed in DID document.

property type: *aries_cloudagent.connections.models.diddoc.publickey.PublicKeyType*
Accessor for the public key type.

property value: str
Accessor for the public key value.

class aries_cloudagent.connections.models.diddoc.**PublicKeyType**(*value*)

Bases: enum.Enum

Class encapsulating public key types.

ED25519_SIG_2018 = LinkedDataKeySpec(ver_type='Ed25519VerificationKey2018', authn_type='Ed25519SignatureAuthentication2018', specifier='publicKeyBase58')

EDDSA_SA_SIG_SECP256K1 = LinkedDataKeySpec(ver_type='Secp256k1VerificationKey2018', authn_type='Secp256k1SignatureAuthenticationKey2018', specifier='publicKeyHex')

RSA_SIG_2018 = LinkedDataKeySpec(ver_type='RsaVerificationKey2018', authn_type='RsaSignatureAuthentication2018', specifier='publicKeyPem')

property authn_type: str
Accessor for the authentication type identifier.

static get(*val: str*) → *aries_cloudagent.connections.models.diddoc.publickey.PublicKeyType*
Find enum instance corresponding to input value (RsaVerificationKey2018 etc).

Parameters **val** – input value marking public key type

Returns: the public key type

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

property specifier: str
Accessor for the value specifier.

property ver_type: str
Accessor for the verification type identifier.

class aries_cloudagent.connections.models.diddoc.**Service**(*did: str*, *ident: str*, *typ: str*, *recip_keys: Union[Sequence, aries_cloudagent.connections.models.diddoc.publickey.Publi routing_keys: Union[Sequence, aries_cloudagent.connections.models.diddoc.publickey.Publi 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).

property did: str
Accessor for the DID value.

property endpoint: str
Accessor for the endpoint value.

property id: str
Accessor for the service identifier.

property priority: int
Accessor for the priority value.

property recip_keys:
List[*aries_cloudagent.connections.models.diddoc.publickey.PublicKey*]
Accessor for the recipient keys.

property routing_keys:
List[*aries_cloudagent.connections.models.diddoc.publickey.PublicKey*]
Accessor for the routing keys.

to_dict() → dict
Return dict representation of service to embed in DID document.

property type: str
Accessor for the service type.

Submodules

aries_cloudagent.connections.models.diddoc.diddoc module

DID Document classes.

class aries_cloudagent.connections.models.diddoc.diddoc.**DIDDoc**(*did: Optional[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

property authnkey: dict
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

property did: str
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

property pubkey: dict
Accessor for public keys by identifier.

serialize() → dict
Dump current object to a JSON-compatible dictionary.

Returns dict representation of current DIDDoc

property service: dict
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

property authn_type
Alias for field number 1

property specifier
Alias for field number 2

property ver_type
Alias for field number 0

class aries_cloudagent.connections.models.diddoc.publickey.**PublicKey**(*did: str*, *ident: str*, *value: str*, *pk_type: Optional[aries_cloudagent.connections.models.* = *None*, *controller: Optional[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).

property authn: bool
Accessor for the authentication marker.

Returns: whether public key is marked as having DID authentication privilege

property controller: str
Accessor for the controller DID.

property did: str
Accessor for the DID.

property id: str
Accessor for the public key identifier.

to_dict() → dict
Return dict representation of public key to embed in DID document.

property type: *aries_cloudagent.connections.models.diddoc.publickey.PublicKeyType*
Accessor for the public key type.

property value: str
Accessor for the public key value.

class aries_cloudagent.connections.models.diddoc.publickey.**PublicKeyType**(*value*)
Bases: enum.Enum

Class encapsulating public key types.

ED25519_SIG_2018 = LinkedDataKeySpec(ver_type='Ed25519VerificationKey2018', authn_type='Ed25519SignatureAuthentication2018', specifier='publicKeyBase58')

EDDSA_SA_SIG_SECP256K1 = LinkedDataKeySpec(ver_type='Secp256k1VerificationKey2018', authn_type='Secp256k1SignatureAuthenticationKey2018', specifier='publicKeyHex')

RSA_SIG_2018 = LinkedDataKeySpec(ver_type='RsaVerificationKey2018', authn_type='RsaSignatureAuthentication2018', specifier='publicKeyPem')

property authn_type: str
Accessor for the authentication type identifier.

static get(*val: str*) → *aries_cloudagent.connections.models.diddoc.publickey.PublicKeyType*
Find enum instance corresponding to input value (RsaVerificationKey2018 etc).

Parameters **val** – input value marking public key type

Returns: the public key type

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

property specifier: str
Accessor for the value specifier.

property ver_type: str
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,* aries_cloudagent.connections.models.diddoc.publi *routing_keys: Union[Sequence,* aries_cloudagent.connections.models.diddoc.publi *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).

property did: str
Accessor for the DID value.

property endpoint: str
Accessor for the endpoint value.

property id: str
Accessor for the service identifier.

property priority: int
Accessor for the priority value.

property recip_keys:
List[*aries_cloudagent.connections.models.diddoc.publickey.PublicKey*]
Accessor for the recipient keys.

property routing_keys:
List[*aries_cloudagent.connections.models.diddoc.publickey.PublicKey*]
Accessor for the routing keys.

to_dict() → dict
Return dict representation of service to embed in DID document.

property type: str
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: Optional[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: Optional[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.conn_record module

Handle connection information interface with non-secrets storage.

class aries_cloudagent.connections.models.conn_record.**ConnRecord**(*, *connection_id: Optional[str] = None*, *my_did: Optional[str] = None*, *their_did: Optional[str] = None*, *their_label: Optional[str] = None*, *their_role: Optional[Union[str, aries_cloudagent.connections.models.conn_record = None*, *invitation_key: Optional[str] = None*, *invitation_msg_id: Optional[str] = None*, *request_id: Optional[str] = None*, *state: Optional[Union[str, aries_cloudagent.connections.models.conn_record = None*, *inbound_connection_id: Optional[str] = None*, *error_msg: Optional[str] = None*, *routing_state: Optional[str] = None*, *accept: Optional[str] = None*, *invitation_mode: Optional[str] = None*, *alias: Optional[str] = None*, *their_public_did: Optional[str] = None*, *rfc23_state: Optional[str] = None*, *initiator: Optional[str] = None*, *connection_protocol: Optional[Union[str, aries_cloudagent.connections.models.conn_record = None*, ***kwargs*)

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

Represents a single pairwise connection.

ACCEPT_AUTO = 'auto'

ACCEPT_MANUAL = 'manual'

INVITATION_MODE_MULTI = 'multi'

INVITATION_MODE_ONCE = 'once'

INVITATION_MODE_STATIC = 'static'

LOG_STATE_FLAG = 'debug.connections'

class Meta

Bases: object

ConnRecord metadata.

schema_class = 'ConnRecordSchema'

class Protocol(*value*)
Bases: enum.Enum

Supported Protocols for Connection.

RFC_0023 = 'didexchange/1.0'

RFC_0160 = 'connections/1.0'

property aries_protocol
Return used connection protocol.

classmethod get(*label: Union[str,*
aries_cloudagent.connections.models.conn_record.ConnRecord.Protocol*]*)
Get aries protocol enum for label.

RECORD_ID_NAME = 'connection_id'

RECORD_TOPIC: Optional[str] = 'connections'

RECORD_TYPE = 'connection'

RECORD_TYPE_INVITATION = 'connection_invitation'

RECORD_TYPE_METADATA = 'connection_metadata'

RECORD_TYPE_REQUEST = 'connection_request'

ROUTING_STATE_ACTIVE = 'active'

ROUTING_STATE_ERROR = 'error'

ROUTING_STATE_NONE = 'none'

ROUTING_STATE_REQUEST = 'request'

class Role(*value*)
Bases: enum.Enum

RFC 160 (inviter, invitee) = RFC 23 (responder, requester).

REQUESTER = ('invitee', 'requester')

RESPONDER = ('inviter', 'responder')

flip()
Return opposite interlocutor role: theirs for ours, ours for theirs.

classmethod get(*label: Union[str,*
aries_cloudagent.connections.models.conn_record.ConnRecord.Role*]*)
Get role enum for label.

property rfc160
Return RFC 160 (connection protocol) nomenclature.

property rfc23
Return RFC 23 (DID exchange protocol) nomenclature.

class State(*value*)
Bases: enum.Enum

Collator for equivalent states between RFC 160 and RFC 23.

On the connection record, the state has to serve for both RFCs. Hence, internally, RFC23 requester/responder states collate to their RFC160 condensed equivalent.

ABANDONED = ('error', 'abandoned')

COMPLETED = ('active', 'completed')

INIT = ('init', 'start')

INVITATION = ('invitation', 'invitation')

REQUEST = ('request', 'request')

RESPONSE = ('response', 'response')

classmethod get(*label: Union[str,* aries_cloudagent.connections.models.conn_record.ConnRecord.State*]*)
Get state enum for label.

property rfc160
Return RFC 160 (connection protocol) nomenclature.

property rfc23
Return RFC 23 (DID exchange protocol) nomenclature to record logic.

rfc23strict(*their_role:* aries_cloudagent.connections.models.conn_record.ConnRecord.Role)
Return RFC 23 (DID exchange protocol) nomenclature to role as per RFC.

TAG_NAMES = {'invitation_key', 'invitation_msg_id', 'my_did', 'request_id', 'state', 'their_did', 'their_public_did', 'their_role'}

async attach_invitation(*session:* aries_cloudagent.core.profile.ProfileSession, *invitation:* *Union[*aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.ConnectionI aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.InvitationMessage*]*)
Persist the related connection invitation to storage.

Parameters

- **session** – The active profile session
- **invitation** – The invitation to relate to this connection record

async attach_request(*session:* aries_cloudagent.core.profile.ProfileSession, *request:* *Union[*aries_cloudagent.protocols.connections.v1_0.messages.connection_request.ConnectionReques aries_cloudagent.protocols.didexchange.v1_0.messages.request.DIDXRequest*]*)
Persist the related connection request to storage.

Parameters

- **session** – The active profile session
- **request** – The request to relate to this connection record

property connection_id: str
Accessor for the ID associated with this connection.

async delete_record(*session:* aries_cloudagent.core.profile.ProfileSession)
Perform connection record deletion actions.

Parameters **session** (*ProfileSession*) – session

async classmethod find_existing_connection(*session:* aries_cloudagent.core.profile.ProfileSession, *their_public_did: str*) → Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]
Retrieve existing active connection records (public did).

Parameters

- **session** – The active profile session
- **their_public_did** – Inviter public DID

property is_multiuse_invitation: bool
Accessor for multi use invitation mode.

property is_ready: str
Accessor for connection readiness.

async metadata_delete(*session:* aries_cloudagent.core.profile.ProfileSession, *key: str*)
Delete custom metadata associated with this connection.

Parameters

- **session** (*ProfileSession*) – session used for storage
- **key** (*str*) – key of metadata to delete

async metadata_get(*session:* aries_cloudagent.core.profile.ProfileSession, *key: str*, *default: Optional[Any] = None*) → Any
Retrieve arbitrary metadata associated with this connection.

Parameters

- **session** (*ProfileSession*) – session used for storage
- **key** (*str*) – key identifying metadata
- **default** (*Any*) – default value to get; type should be a JSON compatible value.

Returns metadata stored by key

Return type Any

async metadata_get_all(*session:* aries_cloudagent.core.profile.ProfileSession) → dict
Return all custom metadata associated with this connection.

Parameters **session** (*ProfileSession*) – session used for storage

Returns dictionary representation of all metadata values

Return type dict

async metadata_set(*session:* aries_cloudagent.core.profile.ProfileSession, *key: str*, *value: Any*)
Set arbitrary metadata associated with this connection.

Parameters

- **session** (*ProfileSession*) – session used for storage
- **key** (*str*) – key identifying metadata
- **value** (*Any*) – value to set

async post_save(*session:* aries_cloudagent.core.profile.ProfileSession, **args*, ***kwargs*)
Perform post-save actions.

Parameters **session** – The active profile session

property record_value: dict
Accessor to for the JSON record value properties for this connection.

async classmethod retrieve_by_alias(*session:* aries_cloudagent.core.profile.ProfileSession, *alias: str*) → *aries_cloudagent.connections.models.conn_record.ConnRecord*

Retrieve a connection record from an alias.

Parameters

- **session** – The active profile session
- **alias** – The alias of the connection

async classmethod retrieve_by_did(*session:* aries_cloudagent.core.profile.ProfileSession, *their_did: Optional[str] = None*, *my_did: Optional[str] = None*, *their_role: Optional[str] = None*) → *aries_cloudagent.connections.models.conn_record.ConnRecord*

Retrieve a connection record by target DID.

Parameters

- **session** – The active profile session
- **their_did** – The target DID to filter by
- **my_did** – One of our DIDs to filter by
- **my_role** – Filter connections by their role

async classmethod retrieve_by_invitation_key(*session:* aries_cloudagent.core.profile.ProfileSession, *invitation_key: str*, *their_role: Optional[str] = None*) → *aries_cloudagent.connections.models.conn_record.ConnRecord*

Retrieve a connection record by invitation key.

Parameters

- **session** – The active profile session
- **invitation_key** – The key on the originating invitation
- **initiator** – Filter by the initiator value

async classmethod retrieve_by_invitation_msg_id(*session:* aries_cloudagent.core.profile.ProfileSession, *invitation_msg_id: str*, *their_role: Optional[str] = None*) → Optional[*aries_cloudagent.connections.models.conn_record.ConnRe*

Retrieve a connection record by invitation_msg_id.

Parameters

- **session** – The active profile session
- **invitation_msg_id** – Invitation message identifier
- **initiator** – Filter by the initiator value

async classmethod retrieve_by_request_id(*session:* aries_cloudagent.core.profile.ProfileSession, *request_id: str*, *their_role: Optional[str] = None*) → *aries_cloudagent.connections.models.conn_record.ConnRecord*

Retrieve a connection record from our previous request ID.

Parameters

- **session** – The active profile session

- **request_id** – The ID of the originating connection request

async retrieve_invitation(*session:* aries_cloudagent.core.profile.ProfileSession) → Union[*aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.Connecti*
aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.InvitationMessage]

Retrieve the related connection invitation.

Parameters **session** – The active profile session

async retrieve_request(*session:* aries_cloudagent.core.profile.ProfileSession) → Union[*aries_cloudagent.protocols.connections.v1_0.messages.connection_request.ConnectionRequ*
aries_cloudagent.protocols.didexchange.v1_0.messages.request.DIDXRequest]

Retrieve the related connection invitation.

Parameters **session** – The active profile session

property rfc23_state: str

RFC23 state per RFC text, potentially particular to role.

class aries_cloudagent.connections.models.conn_record.**ConnRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema to allow serialization/deserialization of connection records.

class Meta

Bases: object

ConnRecordSchema metadata.

model_class

alias of *aries_cloudagent.connections.models.conn_record.ConnRecord*

accept

alias

connection_id

connection_protocol

error_msg

inbound_connection_id

invitation_key

invitation_mode

invitation_msg_id

my_did

request_id

rfc23_state

routing_state

their_did

their_label

their_public_did

their_role

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: Optional[str] = None*, *endpoint: Optional[str] = None*, *label: Optional[str] = None*, *recipient_keys: Optional[Sequence[str]] = None*, *routing_keys: Optional[Sequence[str]] = None*, *sender_key: Optional[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: Any*, ***kwargs: Any*)

Bases: marshmallow.

ConnectionTarget schema.

class Meta

Bases: object

ConnectionTargetSchema metadata.

model_class

alias of *aries_cloudagent.connections.models.connection_target.ConnectionTarget*

did

endpoint

label

recipient_keys

routing_keys

sender_key

Submodules

aries_cloudagent.connections.base_manager module

Class to provide some common utilities.

For Connection, DIDExchange and OutOfBand Manager.

class aries_cloudagent.connections.base_manager.**BaseConnectionManager**(*profile:* aries_cloudagent.core.profile.Profile)

Bases: object

Class to provide utilities regarding connection_targets.

RECORD_TYPE_DID_DOC = 'did_doc'

RECORD_TYPE_DID_KEY = 'did_key'

SUPPORTED_KEY_TYPES = (pydid.verification_method.Ed25519VerificationKey2018,)

async add_key_for_did(*did: str*, *key: str*)

Store a verkey for lookup against a DID.

Parameters

- **did** – The DID to associate with this key
- **key** – The verkey to be added

async create_did_document(*did_info:* aries_cloudagent.wallet.did_info.DIDInfo, *inbound_connection_id: Optional[str] = None*, *svc_endpoints: Optional[Sequence[str]] = None*, *mediation_records: Optional[List[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.M *= None*) → *aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc*

Create our DID doc for a given DID.

Parameters

- **did_info** – The DID information (DID and verkey) used in the connection
- **inbound_connection_id** – The ID of the inbound routing connection to use
- **svc_endpoints** – Custom endpoints for the DID Document
- **mediation_record** – The record for mediation that contains routing_keys and service endpoint

Returns The prepared *DIDDoc* instance

diddoc_connection_targets(*doc:* aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc, *sender_verkey: str*, *their_label: Optional[str] = None*) → Sequence[*aries_cloudagent.connections.models.connection_target.ConnectionTarget*]

Get a list of connection targets from a DID Document.

Parameters

- **doc** – The DID Document to create the target from
- **sender_verkey** – The verkey we are using
- **their_label** – The connection label they are using

async fetch_connection_targets(*connection:* aries_cloudagent.connections.models.conn_record.ConnRecord) → Sequence[*aries_cloudagent.connections.models.connection_target.ConnectionTarget*]

Get a list of connection targets from a *ConnRecord*.

Parameters **connection** – The connection record (with associated *DIDDoc*) used to generate the connection target

async fetch_did_document(*did: str*) → Tuple[*aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc*, *aries_cloudagent.storage.record.StorageRecord*]

Retrieve a DID Document for a given DID.

Parameters **did** – The DID to search for

async find_did_for_key(*key: str*) → str

Find the DID previously associated with a key.

Parameters **key** – The verkey to look up

async remove_keys_for_did(*did: str*)

Remove all keys associated with a DID.

Parameters **did** – The DID for which to remove keys

async resolve_invitation(*did: str*, *service_accept: Optional[Sequence[str]] = None*)

Resolve invitation with the DID Resolver.

Parameters **did** – Document ID to resolve

async store_did_document(*did_doc:* aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc)

Store a DID document.

Parameters **did_doc** – The *DIDDoc* instance to persist

exception aries_cloudagent.connections.base_manager.**BaseConnectionManagerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

BaseConnectionManager error.

aries_cloudagent.core package

Subpackages

aries_cloudagent.core.in_memory package

In-memory wallet support.

class aries_cloudagent.core.in_memory.**InMemoryProfile**(*, *context: Optional[*aries_cloudagent.config.injection_context.InjectionContex = None*, *name: Optional[str] = None*)

Bases: *aries_cloudagent.core.profile.Profile*

Provide access to in-memory profile management.

Used primarily for testing.

BACKEND_NAME: str = 'in_memory'

TEST_PROFILE_NAME = 'test-profile'

bind_providers()
Initialize the profile-level instance providers.

session(*context: Optional[*aries_cloudagent.config.injection_context.InjectionContext*] = None*) → *aries_cloudagent.core.profile.ProfileSession*
Start a new interactive session with no transaction support requested.

classmethod test_profile(*settings: Optional[Mapping[str, Any]] = None*, *bind: Optional[Mapping[Type, Any]] = None*) → *aries_cloudagent.core.in_memory.profile.InMemoryProfile*
Used in tests to create a standard InMemoryProfile.

classmethod test_session(*settings: Optional[Mapping[str, Any]] = None*, *bind: Optional[Mapping[Type, Any]] = None*) → *aries_cloudagent.core.in_memory.profile.InMemoryProfileSession*
Used in tests to quickly create InMemoryProfileSession.

transaction(*context: Optional[*aries_cloudagent.config.injection_context.InjectionContext*] = None*) → *aries_cloudagent.core.profile.ProfileSession*
Start a new interactive session with commit and rollback support.

If the current backend does not support transactions, then commit and rollback operations of the session will not have any effect.

class aries_cloudagent.core.in_memory.**InMemoryProfileManager**
Bases: *aries_cloudagent.core.profile.ProfileManager*

Manager for producing in-memory wallet/storage implementation.

async open(*context:* aries_cloudagent.config.injection_context.InjectionContext, *config: Optional[Mapping[str, Any]] = None*) → *aries_cloudagent.core.profile.Profile*
Open an instance of an existing profile.

async provision(*context:* aries_cloudagent.config.injection_context.InjectionContext, *config: Optional[Mapping[str, Any]] = None*) → *aries_cloudagent.core.profile.Profile*
Provision a new instance of a profile.

class aries_cloudagent.core.in_memory.**InMemoryProfileSession**(*profile:* aries_cloudagent.core.profile.Profile, **, context: Optional[*aries_cloudagent.config.injection_context.Injecti *= None*, *settings: Optional[Mapping[str, Any]] = None*)
Bases: *aries_cloudagent.core.profile.ProfileSession*

An active connection to the profile management backend.

property storage: *aries_cloudagent.storage.base.BaseStorage*
Get the *BaseStorage* implementation (helper specific to in-memory profile).

property wallet: *aries_cloudagent.wallet.base.BaseWallet*
Get the *BaseWallet* implementation (helper specific to in-memory profile).

Subpackages

aries_cloudagent.core.in_memory.didcomm package

Submodules

aries_cloudagent.core.in_memory.didcomm.derive_1pu module

Functions for performing Key Agreement using ECDH-1PU.

aries_cloudagent.core.in_memory.didcomm.derive_1pu.**derive_1pu**(*ze*, *zs*, *alg*, *apu*, *apv*, *keydatalen*)
Generate shared encryption key from two ECDH shared secrets.

aries_cloudagent.core.in_memory.didcomm.derive_1pu.**derive_receiver_1pu**(*epk*, *sender_pk*, *recip_sk*, *alg*, *apu*, *apv*, *keydatalen*)
Generate two shared secrets (ze, zs).

aries_cloudagent.core.in_memory.didcomm.derive_1pu.**derive_sender_1pu**(*epk*, *sender_sk*, *recip_pk*, *alg*, *apu*, *apv*, *keydatalen*)
Generate two shared secrets (ze, zs).

aries_cloudagent.core.in_memory.didcomm.derive_ecdh module

Functions for performing Key Agreement.

aries_cloudagent.core.in_memory.didcomm.derive_ecdh.**concat_kdf**(*shared_secret: bytes*, *alg: Union[str, bytes]*, *apu: Union[str, bytes]*, *apv: Union[str, bytes]*, *keydatalen: int*)
Generate a shared encryption key from a shared secret.

aries_cloudagent.core.in_memory.didcomm.derive_ecdh.**derive_shared_secret**(*private_key: bytes*, *public_key: bytes*)
Generate a shared secret from keys in byte format.

aries_cloudagent.core.in_memory.didcomm.derive_ecdh.**derive_shared_secret_from_key**(*private_key*, *public_key*)
Generate a shared secret from keys in ecdsa.Keys format.

Submodules

aries_cloudagent.core.in_memory.profile module

Manage in-memory profile interaction.

class aries_cloudagent.core.in_memory.profile.**InMemoryProfile**(*, *context: Optional[*aries_cloudagent.config.injection_context.Injec *= None*, *name: Optional[str] = None*)
Bases: *aries_cloudagent.core.profile.Profile*

Provide access to in-memory profile management.

Used primarily for testing.

BACKEND_NAME: str = 'in_memory'

TEST_PROFILE_NAME = 'test-profile'

bind_providers()
Initialize the profile-level instance providers.

session(*context: Optional[*aries_cloudagent.config.injection_context.InjectionContext*] = None*) → *aries_cloudagent.core.profile.ProfileSession*
Start a new interactive session with no transaction support requested.

classmethod test_profile(*settings: Optional[Mapping[str, Any]] = None*, *bind: Optional[Mapping[Type, Any]] = None*) → *aries_cloudagent.core.in_memory.profile.InMemoryProfile*
Used in tests to create a standard InMemoryProfile.

classmethod test_session(*settings: Optional[Mapping[str, Any]] = None*, *bind: Optional[Mapping[Type, Any]] = None*) → *aries_cloudagent.core.in_memory.profile.InMemoryProfileSession*
Used in tests to quickly create InMemoryProfileSession.

transaction(*context: Optional[*aries_cloudagent.config.injection_context.InjectionContext*] = None*) → *aries_cloudagent.core.profile.ProfileSession*
Start a new interactive session with commit and rollback support.

If the current backend does not support transactions, then commit and rollback operations of the session will not have any effect.

class aries_cloudagent.core.in_memory.profile.**InMemoryProfileManager**
Bases: *aries_cloudagent.core.profile.ProfileManager*

Manager for producing in-memory wallet/storage implementation.

async open(*context:* aries_cloudagent.config.injection_context.InjectionContext, *config: Optional[Mapping[str, Any]] = None*) → *aries_cloudagent.core.profile.Profile*
Open an instance of an existing profile.

async provision(*context:* aries_cloudagent.config.injection_context.InjectionContext, *config: Optional[Mapping[str, Any]] = None*) → *aries_cloudagent.core.profile.Profile*
Provision a new instance of a profile.

class aries_cloudagent.core.in_memory.profile.**InMemoryProfileSession**(*profile:* aries_cloudagent.core.profile.Profile, **, context: Optional[*aries_cloudagent.config.injection_con *= None*, *settings: Optional[Mapping[*str, *Any]] = None*)
Bases: *aries_cloudagent.core.profile.ProfileSession*

An active connection to the profile management backend.

property storage: *aries_cloudagent.storage.base.BaseStorage*
Get the *BaseStorage* implementation (helper specific to in-memory profile).

property wallet: *aries_cloudagent.wallet.base.BaseWallet*
Get the *BaseWallet* implementation (helper specific to in-memory profile).

Submodules

aries_cloudagent.core.conductor module

aries_cloudagent.core.dispatcher module

aries_cloudagent.core.error module

Common exception classes.

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

Bases: Exception

Generic exception class which other exceptions should inherit from.

property message: str

Accessor for the error message.

property roll_up: str

Accessor for nested error messages rolled into one line.

For display: aiohttp.web errors truncate after newline.

exception aries_cloudagent.core.error.**ProfileDuplicateError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.ProfileError*

Profile with the given name already exists.

exception aries_cloudagent.core.error.**ProfileError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base error for profile operations.

exception aries_cloudagent.core.error.**ProfileNotFoundError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.ProfileError*

Requested profile was not found.

exception aries_cloudagent.core.error.**ProfileSessionInactiveError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.ProfileError*

Error raised when a profile session is not currently active.

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

Bases: *aries_cloudagent.core.error.BaseError*

Error raised when there is a problem starting the conductor.

aries_cloudagent.core.event_bus module

A simple event bus.

class aries_cloudagent.core.event_bus.**Event**(*topic: str*, *payload: Optional[Any] = None*)

Bases: object

A simple event object.

property payload

Return this event's payload.

property topic

Return this event's topic.

with_metadata(*metadata:* aries_cloudagent.core.event_bus.EventMetadata) → *aries_cloudagent.core.event_bus.EventWithMetadata*

Annotate event with metadata and return EventWithMetadata object.

class aries_cloudagent.core.event_bus.**EventBus**

Bases: object

A simple event bus implementation.

async notify(*profile: Profile*, *event:* aries_cloudagent.core.event_bus.Event)

Notify subscribers of event.

Parameters

- **profile** (*Profile*) – context of the event
- **event** (Event) – event to emit

subscribe(*pattern: Pattern*, *processor: Callable*)

Subscribe to an event.

Parameters

- **pattern** (*Pattern*) – compiled regular expression for matching topics
- **processor** (*Callable*) – async callable accepting profile and event

unsubscribe(*pattern: Pattern*, *processor: Callable*)

Unsubscribe from an event.

This method is idempotent. Repeated calls to unsubscribe will not result in errors.

Parameters

- **pattern** (*Pattern*) – regular expression used to subscribe the processor
- **processor** (*Callable*) – processor to unsubscribe

wait_for_event(*waiting_profile: Profile*, *pattern: Pattern*, *cond: Optional[Callable[[*aries_cloudagent.core.event_bus.Event*], bool]] = None*) → Iterator[Awaitable[*aries_cloudagent.core.event_bus.Event*]]

Capture an event and retrieve its value.

class aries_cloudagent.core.event_bus.**EventMetadata**(*pattern: Pattern*, *match: Match[str]*)

Bases: tuple

Metadata passed alongside events to add context.

property match

Alias for field number 1

property pattern

Alias for field number 0

class aries_cloudagent.core.event_bus.**EventWithMetadata**(*topic: str*, *payload: Any*, *metadata:* aries_cloudagent.core.event_bus.EventMetadata)

Bases: *aries_cloudagent.core.event_bus.Event*

Event with metadata passed alongside events to add context.

property metadata: *aries_cloudagent.core.event_bus.EventMetadata*

Return metadata.

class aries_cloudagent.core.event_bus.**MockEventBus**

Bases: *aries_cloudagent.core.event_bus.EventBus*

A mock EventBus for testing.

async notify(*profile: Profile*, *event:* aries_cloudagent.core.event_bus.Event)

Append the event to MockEventBus.events.

aries_cloudagent.core.goal_code_registry module

Handle registration and publication of supported goal codes.

class aries_cloudagent.core.goal_code_registry.**GoalCodeRegistry**

Bases: object

Goal code registry.

goal_codes_matching_query(*query: str*) → Sequence[str]

Return a list of goal codes matching a query string.

register_controllers(**controller_sets*)

Add new controllers.

Parameters **controller_sets** – Mappings of controller to coroutines

aries_cloudagent.core.oob_processor module

Oob message processor and functions.

class aries_cloudagent.core.oob_processor.**OobMessageProcessor**(*inbound_message_router: Callable[[*aries_cloudagent.core.profile.Profile, aries_cloudagent.transport.inbound.message.InboundM *Optional[bool]], None]*)

Bases: object

Out of band message processor.

async clean_finished_oob_record(*profile:* aries_cloudagent.core.profile.Profile, *message:* aries_cloudagent.messaging.agent_message.AgentMessage)

Clean up oob record associated with agent message, if applicable.

async find_oob_record_for_inbound_message(*context:* aries_cloudagent.messaging.request_context.RequestContext) → Optional[*aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record.O*

Find oob record for inbound message.

async find_oob_target_for_outbound_message(*profile:* aries_cloudagent.core.profile.Profile, *outbound_message:* aries_cloudagent.transport.outbound.message.OutboundMessage) → Optional[*aries_cloudagent.connections.models.connection_target.Connecti*

Find connection target for the outbound message.

get_thread_id(*message: Dict[str, Any]*) → str

Extract thread id from agent message dict.

async handle_message(*profile:* aries_cloudagent.core.profile.Profile, *messages: List[Dict[str, Any]]*, *oob_record:* aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record.OobRecord, *their_service: Optional[*aries_cloudagent.messaging.decorators.service_decorator.ServiceDecorator*] = None*)

Message handler for inbound messages.

exception aries_cloudagent.core.oob_processor.**OobMessageProcessorError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base error for OobMessageProcessor.

aries_cloudagent.core.plugin_registry module

Handle registration of plugin modules for extending functionality.

class aries_cloudagent.core.plugin_registry.**PluginRegistry**(*blocklist: Iterable[str] = []*)

Bases: object

Plugin registry for indexing application plugins.

async init_context(*context:* aries_cloudagent.config.injection_context.InjectionContext)

Call plugin setup methods on the current context.

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

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

property plugin_names: Sequence[str]
Accessor for a list of all plugin modules.

property plugins: Sequence[module]
Accessor for a list of all plugin modules.

post_process_routes(*app*)
Call route binary file response OpenAPI fixups if applicable.

async 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.

register_protocol_events(*context:* aries_cloudagent.config.injection_context.InjectionContext)
Call route register_events methods on the current context.

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

aries_cloudagent.core.profile module

Classes for managing profile information within a request context.

class aries_cloudagent.core.profile.**Profile**(*, *context: Optional[*aries_cloudagent.config.injection_context.InjectionContext*] = None*, *name: Optional[str] = None*, *created: bool = False*)

Bases: abc.ABC

Base abstraction for handling identity-related state.

BACKEND_NAME: str = None

DEFAULT_NAME: str = 'default'

property backend: str
Accessor for the backend implementation name.

async close()
Close the profile instance.

property context: *aries_cloudagent.config.injection_context.InjectionContext*
Accessor for the injection context.

property created: bool
Accessor for the created flag indicating a new profile.

inject(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*) → aries_cloudagent.config.base.InjectType
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

inject_or(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*, *default: Optional[aries_cloudagent.config.base.InjectType] = None*) → Optional[aries_cloudagent.config.base.InjectType]
Get the provided instance of a given class identifier or default if not found.

Parameters

- **base_cls** – The base class to retrieve an instance of
- **settings** – An optional dict providing configuration to the provider
- **default** – default return value if no instance is found

Returns An instance of the base class, or None

property name: str
Accessor for the profile name.

async notify(*topic: str*, *payload: Any*)
Signal an event.

async remove()
Remove the profile.

abstract session(*context: Optional[aries_cloudagent.config.injection_context.InjectionContext] = None*) → *aries_cloudagent.core.profile.ProfileSession*
Start a new interactive session with no transaction support requested.

property settings: *aries_cloudagent.config.base.BaseSettings*
Accessor for scope-specific settings.

abstract transaction(*context: Optional[aries_cloudagent.config.injection_context.InjectionContext] = None*) → *aries_cloudagent.core.profile.ProfileSession*
Start a new interactive session with commit and rollback support.

If the current backend does not support transactions, then commit and rollback operations of the session will not have any effect.

class aries_cloudagent.core.profile.**ProfileManager**
Bases: abc.ABC

Handle provision and open for profile instances.

abstract async open(*context:* aries_cloudagent.config.injection_context.InjectionContext, *config: Optional[Mapping[str, Any]] = None*) → *aries_cloudagent.core.profile.Profile*
Open an instance of an existing profile.

abstract async provision(*context:* aries_cloudagent.config.injection_context.InjectionContext, *config: Optional[Mapping[str, Any]] = None*) → *aries_cloudagent.core.profile.Profile*
Provision a new instance of a profile.

class aries_cloudagent.core.profile.**ProfileManagerProvider**
Bases: *aries_cloudagent.config.base.BaseProvider*

The standard profile manager provider which keys off the selected wallet type.

MANAGER_TYPES = {'askar': 'aries_cloudagent.askar.profile.AskarProfileManager', 'in_memory': 'aries_cloudagent.core.in_memory.InMemoryProfileManager', 'indy': 'aries_cloudagent.indy.sdk.profile.IndySdkProfileManager'}

provide(*settings:* aries_cloudagent.config.base.BaseSettings, *injector:* aries_cloudagent.config.base.BaseInjector)
Create the profile manager instance.

class aries_cloudagent.core.profile.**ProfileSession**(*profile:* aries_cloudagent.core.profile.Profile, *, *context: Optional[*aries_cloudagent.config.injection_context.InjectionContext*] = None*, *settings: Optional[Mapping[str, Any]] = None*)
Bases: abc.ABC

An active connection to the profile management backend.

property active: bool
Accessor for the session active state.

async commit()
Commit any updates performed within the transaction.

If the current session is not a transaction, then nothing is performed.

property context: *aries_cloudagent.config.injection_context.InjectionContext*
Accessor for the associated injection context.

inject(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*) → aries_cloudagent.config.base.InjectType
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

inject_or(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*, *default: Optional[aries_cloudagent.config.base.InjectType] = None*) → Optional[aries_cloudagent.config.base.InjectType]
Get the provided instance of a given class identifier or default if not found.

Parameters

- **base_cls** – The base class to retrieve an instance of
- **settings** – An optional dict providing configuration to the provider
- **default** – default return value if no instance is found

Returns An instance of the base class, or None

property is_transaction: bool
Check if the session supports commit and rollback operations.

property profile: *aries_cloudagent.core.profile.Profile*
Accessor for the associated profile instance.

async rollback()
Roll back any updates performed within the transaction.

If the current session is not a transaction, then nothing is performed.

property settings: *aries_cloudagent.config.base.BaseSettings*
Accessor for scope-specific settings.

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.

property controllers: Mapping[str, str]
Accessor for a list of all protocol controller functions.

create_msg_types_for_minor_version(*typesets*, *version_definition*)
Return mapping of message type to module path for minor versions.

Parameters

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

Returns Typesets mapping

property message_types: Sequence[str]
Accessor for a list of all message types.

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

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

property protocols: Sequence[str]
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

aries_cloudagent.core.util module

Core utilities and constants.

aries_cloudagent.core.util.**get_proto_default_version**(*def_path: str*, *major_version: int = 1*) → str
Return default protocol version from version_definition.

async aries_cloudagent.core.util.**get_proto_default_version_from_msg_class**(*profile:* aries_cloudagent.core.profile.Profile, *msg_class: type*, *major_version: int = 1*) → str
Return default protocol version from version_definition.

async aries_cloudagent.core.util.**get_version_def_from_msg_class**(*profile:* aries_cloudagent.core.profile.Profile, *msg_class: type*, *major_version: int = 1*)
Return version_definition of a protocol from msg_class.

aries_cloudagent.core.util.**get_version_from_message**(*msg:* aries_cloudagent.messaging.agent_message.AgentMessage) → str
Return version from provided AgentMessage.

aries_cloudagent.core.util.**get_version_from_message_type**(*msg_type: str*) → str
Return version from provided message_type.

async aries_cloudagent.core.util.**validate_get_response_version**(*profile:* aries_cloudagent.core.profile.Profile, *rec_version: str*, *msg_class: type*) → Tuple[str, Optional[str]]
Return a tuple with version to respond with and warnings.

Process received version and protocol version definition, returns the tuple.

Parameters

- **profile** – Profile
- **rec_version** – received version from message
- **msg_class** – type

Returns Tuple with response version and any warnings

aries_cloudagent.did package

Submodules

aries_cloudagent.did.did_key module

DID Key class and resolver methods.

class aries_cloudagent.did.did_key.**DIDKey**(*public_key: bytes*, *key_type:* aries_cloudagent.wallet.key_type.KeyType)

Bases: object

DID Key parser and resolver.

property did: str
key string.

Type Getter for full did

property did_doc: dict
key.

Type Getter for did document associated with did

property fingerprint: str
Getter for did key fingerprint.

classmethod from_did(*did: str*) → *aries_cloudagent.did.did_key.DIDKey*
Initialize a new DIDKey instance from a fully qualified did:key string.

Extracts the fingerprint from the did:key and uses that to constrcut the did:key.

classmethod from_fingerprint(*fingerprint: str*, *key_types=None*) → *aries_cloudagent.did.did_key.DIDKey*
Initialize new DIDKey instance from multibase encoded fingerprint.

The fingerprint contains both the public key and key type.

classmethod from_public_key(*public_key: bytes*, *key_type:* aries_cloudagent.wallet.key_type.KeyType) → *aries_cloudagent.did.did_key.DIDKey*
Initialize new DIDKey instance from public key and key type.

classmethod from_public_key_b58(*public_key: str*, *key_type:* aries_cloudagent.wallet.key_type.KeyType) → *aries_cloudagent.did.did_key.DIDKey*
Initialize new DIDKey instance from base58 encoded public key and key type.

property key_id: str
Getter for key id.

property key_type: *aries_cloudagent.wallet.key_type.KeyType*
Getter for key type.

property prefixed_public_key: bytes
Getter for multicodec prefixed public key.

property public_key: bytes
Getter for public key.

property public_key_b58: str
Getter for base58 encoded public key.

aries_cloudagent.did.did_key.**construct_did_key_bls12381g1**(*did_key:* aries_cloudagent.did.did_key.DIDKey) → dict

Construct BLS12381G1 did:key.

Parameters **did_key** (DIDKey) – did key instance to parse bls12381g1 did:key document from

Returns The bls12381g1 did:key did document

Return type dict

aries_cloudagent.did.did_key.**construct_did_key_bls12381g1g2**(*did_key:* aries_cloudagent.did.did_key.DIDKey) → dict

Construct BLS12381G1G2 did:key.

Parameters **did_key** (DIDKey) – did key instance to parse bls12381g1g2 did:key document from

Returns The bls12381g1g2 did:key did document

Return type dict

aries_cloudagent.did.did_key.**construct_did_key_bls12381g2**(*did_key:* aries_cloudagent.did.did_key.DIDKey) → dict

Construct BLS12381G2 did:key.

Parameters **did_key** (DIDKey) – did key instance to parse bls12381g2 did:key document from

Returns The bls12381g2 did:key did document

Return type dict

aries_cloudagent.did.did_key.**construct_did_key_ed25519**(*did_key:* aries_cloudagent.did.did_key.DIDKey) → dict

Construct Ed25519 did:key.

Parameters **did_key** (DIDKey) – did key instance to parse ed25519 did:key document from

Returns The ed25519 did:key did document

Return type dict

aries_cloudagent.did.did_key.**construct_did_key_x25519**(*did_key:* aries_cloudagent.did.did_key.DIDKey) → dict

Construct X25519 did:key.

Parameters **did_key** (DIDKey) – did key instance to parse x25519 did:key document from

Returns The x25519 did:key did document

Return type dict

aries_cloudagent.did.did_key.**construct_did_signature_key_base**(**, id: str, key_id: str, verification_method: dict*)

Create base did key structure to use for most signature keys.

May not be suitable for all did key types

aries_cloudagent.holder package

Submodules

aries_cloudagent.holder.routes module

Holder admin routes.

class aries_cloudagent.holder.routes.**AttributeMimeTypesResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for credential attribute MIME type.

results

class aries_cloudagent.holder.routes.**CredInfoListSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for credential query.

results

class aries_cloudagent.holder.routes.**CredRevokedQueryStringSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters and validators for request seeking cred revocation status.

fro

to

class aries_cloudagent.holder.routes.**CredRevokedResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for credential revoked request.

revoked

class aries_cloudagent.holder.routes.**CredentialsListQueryStringSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Parameters and validators for query string in credentials list query.

count

start

wql

class aries_cloudagent.holder.routes.**HolderCredIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters and validators for request taking credential id.

credential_id

class aries_cloudagent.holder.routes.**HolderModuleResponseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Response schema for Holder Module.

class aries_cloudagent.holder.routes.**VCRecordListSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Result schema for W3C credential query.

results

class aries_cloudagent.holder.routes.**W3CCredentialsListRequestSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Parameters and validators for W3C credentials request.

contexts

given_id

issuer_id

max_results

proof_types

schema_ids

subject_ids

tag_query

types

aries_cloudagent.holder.routes.**post_process_routes**(*app: aiohttp.web.Application*)
Amend swagger API.

async aries_cloudagent.holder.routes.**register**(*app: aiohttp.web.Application*)
Register routes.

aries_cloudagent.indy package

Subpackages

aries_cloudagent.indy.credx package

Submodules

aries_cloudagent.indy.credx.holder module

aries_cloudagent.indy.credx.issuer module

aries_cloudagent.indy.credx.verifier module

aries_cloudagent.indy.models package

Submodules

aries_cloudagent.indy.models.cred module

Credential artifacts.

class aries_cloudagent.indy.models.cred.**IndyAttrValue**(*raw: Optional[str] = None*, *encoded: Optional[str] = None*, ***kwargs*)

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

Indy attribute value.

class Meta

Bases: object

Indy attribute value.

schema_class = 'IndyAttrValueSchema'

class aries_cloudagent.indy.models.cred.**IndyAttrValueSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy attribute value schema.

class Meta

Bases: object

Indy attribute value schema metadata.

model_class

alias of *aries_cloudagent.indy.models.cred.IndyAttrValue*

encoded

raw

class aries_cloudagent.indy.models.cred.**IndyCredential**(*schema_id: Optional[str] = None*, *cred_def_id: Optional[str] = None*, *rev_reg_id: Optional[str] = None*, *values: Optional[Mapping[str, aries_cloudagent.indy.models.cred.IndyAttrValue]] = None*, *signature: Optional[Mapping] = None*, *signature_correctness_proof: Optional[Mapping] = None*, *rev_reg: Optional[Mapping] = None*, *witness: Optional[Mapping] = None*)

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

Indy credential.

class Meta

Bases: object

Indy credential metadata.

schema_class = 'IndyCredentialSchema'

class aries_cloudagent.indy.models.cred.**IndyCredentialSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy credential schema.

class Meta

Bases: object

Indy credential schemametadata.

model_class

alias of *aries_cloudagent.indy.models.cred.IndyCredential*

cred_def_id

rev_reg

rev_reg_id

schema_id

signature

signature_correctness_proof

values

witness

aries_cloudagent.indy.models.cred_abstract module

Cred abstract artifacts to attach to RFC 453 messages.

class aries_cloudagent.indy.models.cred_abstract.**IndyCredAbstract**(*schema_id: Optional[str] = None*, *cred_def_id: Optional[str] = None*, *nonce: Optional[str] = None*, *key_correctness_proof: Optional[str] = None*, ***kwargs*)

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

Indy credential abstract.

class Meta

Bases: object

Indy credential abstract metadata.

schema_class = 'IndyCredAbstractSchema'

class aries_cloudagent.indy.models.cred_abstract.**IndyCredAbstractSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy credential abstract schema.

class Meta

Bases: object

Indy credential abstract schema metadata.

model_class

alias of *aries_cloudagent.indy.models.cred_abstract.IndyCredAbstract*

cred_def_id

key_correctness_proof

nonce

schema_id

class aries_cloudagent.indy.models.cred_abstract.**IndyKeyCorrectnessProof**(*c: Optional[str] = None*, *xz_cap: Optional[str] = None*, *xr_cap: Optional[Sequence[Sequence[str]]] = None*, ***kwargs*)

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

Indy key correctness proof.

class Meta

Bases: object

IndyKeyCorrectnessProof metadata.

schema_class = 'IndyKeyCorrectnessProofSchema'

class aries_cloudagent.indy.models.cred_abstract.**IndyKeyCorrectnessProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy key correctness proof schema.

class Meta

Bases: object

Indy key correctness proof schema metadata.

model_class

alias of *aries_cloudagent.indy.models.cred_abstract.IndyKeyCorrectnessProof*

c

xr_cap

xz_cap

aries_cloudagent.indy.models.cred_def module

Schema artifacts.

class aries_cloudagent.indy.models.cred_def.**CredDefValuePrimarySchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Cred def value primary schema.

n

r

rctxt

s

z

class aries_cloudagent.indy.models.cred_def.**CredDefValueRevocationSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Cred def value revocation schema.

g

g_dash

h

h0

h1

h2

h_cap

htilde

pk

u

y

class aries_cloudagent.indy.models.cred_def.**CredDefValueSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Cred def value schema.

primary

revocation

class aries_cloudagent.indy.models.cred_def.**CredentialDefinitionSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Marshmallow schema for indy cred def.

ident

schemaId

tag

typ

value

ver

aries_cloudagent.indy.models.cred_precis module

Admin routes for presentations.

class aries_cloudagent.indy.models.cred_precis.**IndyCredInfo**(*referent: Optional[str] = None*, *attrs: Optional[Mapping] = None*, *schema_id: Optional[str] = None*, *cred_def_id: Optional[str] = None*, *rev_reg_id: Optional[str] = None*, *cred_rev_id: Optional[str] = None*)

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

Indy cred info, as holder gets via indy-sdk.

class Meta
Bases: object

IndyCredInfo metadata.

schema_class = 'IndyCredInfoSchema'

class aries_cloudagent.indy.models.cred_precis.IndyCredInfoSchema(*args: Any, **kwargs: Any)
Bases: marshmallow.

Schema for indy cred-info.

class Meta
Bases: object

Schema metadata.

model_class
alias of aries_cloudagent.indy.models.cred_precis.IndyCredInfo

attrs

cred_def_id

cred_rev_id

referent

rev_reg_id

schema_id

class aries_cloudagent.indy.models.cred_precis.IndyCredPrecisSchema(*args: Any, **kwargs: Any)
Bases: marshmallow.

Schema for precis that indy credential search returns (and aca-py augments).

cred_info

interval

presentation_referents

aries_cloudagent.indy.models.cred_request module

Cred request artifacts to attach to RFC 453 messages.

class aries_cloudagent.indy.models.cred_request.IndyCredRequest(prover_did: Optional[str] = None, cred_def_id: Optional[str] = None, blinded_ms: Optional[Mapping] = None, blinded_ms_correctness_proof: Optional[Mapping] = None, nonce: Optional[str] = None, **kwargs)
Bases: aries_cloudagent.messaging.models.base.BaseModel

Indy credential request.

class Meta
Bases: object

Indy credential request metadata.

schema_class = 'IndyCredRequestSchema'

class aries_cloudagent.indy.models.cred_request.IndyCredRequestSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Indy credential request schema.

class Meta

Bases: object

Indy credential request schema metadata.

model_class

alias of aries_cloudagent.indy.models.cred_request.IndyCredRequest

blinded_ms

blinded_ms_correctness_proof

cred_def_id

nonce

prover_did

aries_cloudagent.indy.models.non_rev_interval module

Indy non-revocation interval.

class aries_cloudagent.indy.models.non_rev_interval.IndyNonRevocationInterval(fro: Optional[int] = None, to: Optional[int] = None, **kwargs)

Bases: aries_cloudagent.messaging.models.base.BaseModel

Indy non-revocation interval.

class Meta

Bases: object

NonRevocationInterval metadata.

schema_class = 'IndyNonRevocationIntervalSchema'

covers(timestamp: Optional[int] = None) → bool

Whether input timestamp (default now) lies within non-revocation interval.

Parameters **timestamp** – time of interest

Returns whether input time satisfies non-revocation interval

timestamp() → bool

Return a timestamp that the non-revocation interval covers.

class aries_cloudagent.indy.models.non_rev_interval.IndyNonRevocationIntervalSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Schema to allow serialization/deserialization of non-revocation intervals.

class Meta
Bases: object

IndyNonRevocationIntervalSchema metadata.

model_class
alias of *aries_cloudagent.indy.models.non_rev_interval.IndyNonRevocationInterval*

fro

to

aries_cloudagent.indy.models.predicate module

Utilities for dealing with predicates.

class aries_cloudagent.indy.models.predicate.**Predicate**(*value*)
Bases: enum.Enum

Enum for predicate types that indy-sdk supports.

GE = Relation(fortran='GE', wql='$gte', math='>=', yes=<function Predicate.<lambda>>, no=<function Predicate.<lambda>>)

GT = Relation(fortran='GT', wql='$gt', math='>', yes=<function Predicate.<lambda>>, no=<function Predicate.<lambda>>)

LE = Relation(fortran='LE', wql='$lte', math='<=', yes=<function Predicate.<lambda>>, no=<function Predicate.<lambda>>)

LT = Relation(fortran='LT', wql='$lt', math='<', yes=<function Predicate.<lambda>>, no=<function Predicate.<lambda>>)

property fortran: str
Fortran nomenclature.

static get(*relation: str*) → *aries_cloudagent.indy.models.predicate.Predicate*
Return enum instance corresponding to input relation string.

property math: str
Mathematical nomenclature.

static to_int(*value: Any*) → int
Cast a value as its equivalent int for indy predicate argument.

Raise ValueError for any input but int, stringified int, or boolean.

Parameters value – value to coerce

property wql: str
WQL nomenclature.

class aries_cloudagent.indy.models.predicate.**Relation**(*fortran*, *wql*, *math*, *yes*, *no*)
Bases: tuple

property fortran
Alias for field number 0

property math
Alias for field number 2

property no
Alias for field number 4

property wql
Alias for field number 1

property yes
Alias for field number 3

aries_cloudagent.indy.models.pres_preview module

aries_cloudagent.indy.models.proof module

Marshmallow bindings for indy proofs.

class aries_cloudagent.indy.models.proof.**IndyEQProof**(*revealed_attrs: Optional[Mapping[str, str]] = None*, *a_prime: Optional[str] = None*, *e: Optional[str] = None*, *v: Optional[str] = None*, *m: Optional[Mapping[str, str]] = None*, *m2: Optional[str] = None*, ***kwargs*)

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

Equality proof for indy primary proof.

class Meta
Bases: object

Equality proof metadata.

schema_class = 'IndyEQProofMeta'

class aries_cloudagent.indy.models.proof.**IndyEQProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy equality proof schema.

class Meta
Bases: object

Indy equality proof metadata.

model_class
alias of *aries_cloudagent.indy.models.proof.IndyEQProof*

a_prime

e

m

m2

revealed_attrs

v

class aries_cloudagent.indy.models.proof.**IndyGEProof**(*u: Optional[Mapping[str, str]] = None*, *r: Optional[Mapping[str, str]] = None*, *mj: Optional[str] = None*, *alpha: Optional[str] = None*, *t: Optional[Mapping[str, str]] = None*, *predicate: Optional[*aries_cloudagent.indy.models.proof.IndyGEProofPred*] = None*, ***kwargs*)

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

Greater-than-or-equal-to proof for indy primary proof.

class Meta

Bases: object

GE proof metadata.

schema_class = 'IndyGEProofMeta'

class aries_cloudagent.indy.models.proof.**IndyGEProofPred**(*attr_name: Optional[str] = None*, *p_type: Optional[str] = None*, *value: Optional[int] = None*, ***kwargs*)

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

Indy GE proof predicate.

class Meta

Bases: object

Indy GE proof predicate metadata.

schema_class = 'IndyGEProofPredSchema'

class aries_cloudagent.indy.models.proof.**IndyGEProofPredSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy GE proof predicate schema.

class Meta

Bases: object

Indy GE proof predicate metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyGEProofPred*

attr_name

p_type

value

class aries_cloudagent.indy.models.proof.**IndyGEProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy GE proof schema.

class Meta

Bases: object

Indy GE proof schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyGEProof*

alpha

mj

predicate

r

t

u

class aries_cloudagent.indy.models.proof.**IndyNonRevocProof**(*x_list: Optional[Mapping] = None*, *c_list: Optional[Mapping] = None*, ***kwargs*)

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

Indy non-revocation proof.

class Meta

Bases: object

Indy non-revocation proof metadata.

schema_class = 'IndyNonRevocProofSchema'

class aries_cloudagent.indy.models.proof.**IndyNonRevocProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy non-revocation proof schema.

class Meta

Bases: object

Indy non-revocation proof schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyNonRevocProof*

c_list

x_list

class aries_cloudagent.indy.models.proof.**IndyPresSpecSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for indy proof specification to send as presentation.

requested_attributes

requested_predicates

self_attested_attributes

trace

class aries_cloudagent.indy.models.proof.**IndyPrimaryProof**(*eq_proof: Optional[*aries_cloudagent.indy.models.proof.IndyEQProof*] = None*, *ge_proofs: Optional[Sequence[*aries_cloudagent.indy.models.proof.IndyG*] = None*, ***kwargs*)

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

Indy primary proof.

class Meta

Bases: object

Indy primary proof metadata.

schema_class = 'IndyPrimaryProofSchema'

class aries_cloudagent.indy.models.proof.IndyPrimaryProofSchema(*args: Any, **kwargs: Any)
Bases: marshmallow.

Indy primary proof schema.

class Meta
Bases: object

Indy primary proof schema metadata.

model_class
alias of aries_cloudagent.indy.models.proof.IndyPrimaryProof

eq_proof

ge_proofs

class aries_cloudagent.indy.models.proof.IndyProof(proof: Optional[aries_cloudagent.indy.models.proof.IndyProofProof] = None, requested_proof: Optional[aries_cloudagent.indy.models.proof.IndyProofRequestedProo = None, identifiers: Optional[Sequence[aries_cloudagent.indy.models.proof.IndyProofIdent = None, **kwargs)
Bases: aries_cloudagent.messaging.models.base.BaseModel

Indy proof.

class Meta
Bases: object

Indy proof metadata.

schema_class = 'IndyProofSchema'

class aries_cloudagent.indy.models.proof.IndyProofIdentifier(schema_id: Optional[str] = None, cred_def_id: Optional[str] = None, rev_reg_id: Optional[str] = None, timestamp: Optional[int] = None, **kwargs)
Bases: aries_cloudagent.messaging.models.base.BaseModel

Indy proof identifier.

class Meta
Bases: object

Indy proof identifier metadata.

schema_class = 'IndyProofIdentifierSchema'

class aries_cloudagent.indy.models.proof.IndyProofIdentifierSchema(*args: Any, **kwargs: Any)
Bases: marshmallow.

Indy proof identifier schema.

class Meta
Bases: object

Indy proof identifier schema metadata.

model_class
alias of aries_cloudagent.indy.models.proof.IndyProofIdentifier

cred_def_id

rev_reg_id

schema_id

timestamp

class aries_cloudagent.indy.models.proof.**IndyProofProof**(*proofs: Optional[Sequence[*aries_cloudagent.indy.models.proof.IndyProo
= None*, *aggregated_proof: Optional[*aries_cloudagent.indy.models.proof.IndyProofProofAg
= None*, ***kwargs*)

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

Indy proof.proof content.

class Meta

Bases: object

Indy proof.proof content metadata.

schema_class = 'IndyProofProofSchema'

class aries_cloudagent.indy.models.proof.**IndyProofProofAggregatedProof**(*c_hash: Optional[str]
= None*, *c_list: Optional[Sequence[Sequence[int]]]
= None*, ***kwargs*)

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

Indy proof.proof aggregated proof.

class Meta

Bases: object

Indy proof.proof aggregated proof metadata.

schema_class = 'IndyProofProofAggregatedProofSchema'

class aries_cloudagent.indy.models.proof.**IndyProofProofAggregatedProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof.proof aggregated proof schema.

class Meta

Bases: object

Indy proof.proof aggregated proof schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyProofProofAggregatedProof*

c_hash

c_list

class aries_cloudagent.indy.models.proof.**IndyProofProofProofsProof**(*primary_proof: Optional[*aries_cloudagent.indy.models.proof.Indy
= None*, *non_revoc_proof: Optional[*aries_cloudagent.indy.models.proof.Indy
= None*, ***kwargs*)

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

Indy proof.proof.proofs constituent proof.

class Meta

Bases: object

Indy proof.proof.proofs constituent proof schema.

schema_class = 'IndyProofProofProofsProofSchema'

class aries_cloudagent.indy.models.proof.**IndyProofProofProofsProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof.proof.proofs constituent proof schema.

class Meta

Bases: object

Indy proof.proof.proofs constituent proof schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyProofProofProofsProof*

non_revoc_proof

primary_proof

class aries_cloudagent.indy.models.proof.**IndyProofProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof.proof content schema.

class Meta

Bases: object

Indy proof.proof content schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyProofProof*

aggregated_proof

proofs

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProof**(*revealed_attrs: Optional[Mapping[str, aries_cloudagent.indy.models.proof.IndyProofReq = None*, *revealed_attr_groups: Optional[Mapping[str, aries_cloudagent.indy.models.proof.IndyProofReq = None*, *self_attested_attrs: Optional[Mapping] = None*, *unrevealed_attrs: Optional[Mapping] = None*, *predicates: Optional[Mapping[str, aries_cloudagent.indy.models.proof.IndyProofReq = None*, ***kwargs*)

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

Indy proof.requested_proof content.

class Meta

Bases: object

Indy proof.requested_proof content metadata.

schema_class = 'IndyProofRequestedProofSchema'

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProofPredicate**(*sub_proof_index: Optional[int] = None*, ***kwargs*)

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

Indy proof requested proof predicate.

class Meta

Bases: object

Indy proof requested proof requested proof predicate metadata.

schema_class = 'IndyProofRequestedProofPredicateSchema'

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProofPredicateSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof requested prrof predicate schema.

class Meta

Bases: object

Indy proof requested proof requested proof predicate schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyProofRequestedProofPredicate*

sub_proof_index

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProofRevealedAttr**(*sub_proof_index: Optional[int] = None*, ***kwargs*)

Bases: *aries_cloudagent.indy.models.proof.RawEncoded*

Indy proof requested proof revealed attr.

class Meta

Bases: object

Indy proof requested proof revealed attr metadata.

schema_class = 'IndyProofRequestedProofRevealedAttrSchema'

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProofRevealedAttrGroup**(*sub_proof_index: Optional[int] = None*, *values: Optional[Mapping[str, aries_cloudagent.indy.mode = None*, ***kwargs*)

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

Indy proof requested proof revealed attr group.

class Meta

Bases: object

Indy proof requested proof revealed attr group metadata.

schema_class = 'IndyProofRequestedProofRevealedAttrGroupSchema'

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProofRevealedAttrGroupSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof requested proof revealed attr group schema.

class Meta

Bases: object

Indy proof requested proof revealed attr group schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof.IndyProofRequestedProofRevealedAttrGroup*

sub_proof_index

values

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProofRevealedAttrSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof requested proof revealed attr schema.

class Meta

Bases: object

Indy proof requested proof revealed attr schema metadata.

model_class
alias of *aries_cloudagent.indy.models.proof.IndyProofRequestedProofRevealedAttr*

sub_proof_index

class aries_cloudagent.indy.models.proof.**IndyProofRequestedProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof requested proof schema.

class Meta
Bases: object

Indy proof requested proof schema metadata.

model_class
alias of *aries_cloudagent.indy.models.proof.IndyProofRequestedProof*

predicates

revealed_attr_groups

revealed_attrs

self_attested_attrs

unrevealed_attrs

class aries_cloudagent.indy.models.proof.**IndyProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy proof schema.

class Meta
Bases: object

Indy proof schema metadata.

model_class
alias of *aries_cloudagent.indy.models.proof.IndyProof*

identifiers

proof

requested_proof

class aries_cloudagent.indy.models.proof.**RawEncoded**(*raw: Optional[str] = None*, *encoded: Optional[str] = None*, ***kwargs*)

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

Raw and encoded attribute values.

class Meta
Bases: object

Raw and encoded attribute values metadata.

schema_class = 'RawEncodedSchema'

class aries_cloudagent.indy.models.proof.**RawEncodedSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Raw and encoded attribute values schema.

class Meta
Bases: object

Raw and encoded attribute values schema metadata.

model_class
alias of aries_cloudagent.indy.models.proof.RawEncoded

encoded

raw

aries_cloudagent.indy.models.proof_request module

Utilities to deal with indy.

class aries_cloudagent.indy.models.proof_request.IndyProofReqAttrSpecSchema(*args: Any, **kwargs: Any)
Bases: marshmallow.

Schema for attribute specification in indy proof request.

name

names

non_revoked

restrictions

validate_fields(data, **kwargs)
Validate schema fields.

Data must have exactly one of name or names; if names then restrictions are mandatory.

Parameters data – The data to validate

Raises ValidationError – if data has both or neither of name and names

class aries_cloudagent.indy.models.proof_request.IndyProofReqPredSpecSchema(*args: Any, **kwargs: Any)
Bases: marshmallow.

Schema for predicate specification in indy proof request.

name

non_revoked

p_type

p_value

restrictions

class aries_cloudagent.indy.models.proof_request.**IndyProofRequest**(*nonce: Optional[str] = None*, *name: Optional[str] = None*, *version: Optional[str] = None*, *requested_attributes: Optional[Mapping] = None*, *requested_predicates: Optional[Mapping] = None*, *non_revoked: Optional[Mapping] = None*, ***kwargs*)

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

Indy proof request.

class Meta

Bases: object

Indy proof request metadata.

schema_class = 'IndyProofRequestSchema'

class aries_cloudagent.indy.models.proof_request.**IndyProofRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for indy proof request.

class Meta

Bases: object

Indy proof request schema metadata.

model_class

alias of *aries_cloudagent.indy.models.proof_request.IndyProofRequest*

name

non_revoked

nonce

requested_attributes

requested_predicates

version

aries_cloudagent.indy.models.requested_creds module

Admin routes for presentations.

class aries_cloudagent.indy.models.requested_creds.**IndyRequestedCredsRequestedAttrSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for requested attributes within indy requested credentials structure.

cred_id

revealed

class aries_cloudagent.indy.models.requested_creds.**IndyRequestedCredsRequestedPredSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for requested predicates within indy requested credentials structure.

cred_id

timestamp

aries_cloudagent.indy.models.revocation module

Revocation artifacts.

class aries_cloudagent.indy.models.revocation.**IndyRevRegDef**(*ver: Optional[str] = None*, *id_: Optional[str] = None*, *revoc_def_type: Optional[str] = None*, *tag: Optional[str] = None*, *cred_def_id: Optional[str] = None*, *value: Optional[*aries_cloudagent.indy.models.revocation.IndyRev *= None*)

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

Indy revocation registry definition.

class Meta

Bases: object

Model metadata.

schema_class = 'IndyRevRegDefSchema'

class aries_cloudagent.indy.models.revocation.**IndyRevRegDefSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy revocation registry definition schema.

class Meta

Bases: object

Schema metadata.

model_class

alias of *aries_cloudagent.indy.models.revocation.IndyRevRegDef*

cred_def_id

id_

revoc_def_type

tag

value

ver

class aries_cloudagent.indy.models.revocation.**IndyRevRegDefValue**(*issuance_type: Optional[str] = None*, *max_cred_num: Optional[int] = None*, *public_keys: Optional[aries_cloudagent.indy.models.revocation.In = None*, *tails_hash: Optional[str] = None*, *tails_location: Optional[str] = None*)

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

Indy revocation registry definition value.

class Meta

Bases: object

Model metadata.

schema_class = 'IndyRevRegDefValueSchema'

class aries_cloudagent.indy.models.revocation.**IndyRevRegDefValuePublicKeys**(*accum_key: Optional[aries_cloudagent.indy.models. = None*)

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

Indy revocation registry definition value public keys.

class Meta

Bases: object

Model metadata.

schema_class = 'IndyRevRegDefValuePublicKeysSchema'

class aries_cloudagent.indy.models.revocation.**IndyRevRegDefValuePublicKeysAccumKey**(*z: Optional[str] = None*)

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

Indy revocation registry definition value public keys accum key.

class Meta

Bases: object

Indy revocation registry definition value public keys accum key metadata.

schema_class = 'IndyRevRegDefValuePublicKeysAccumKeySchema'

class aries_cloudagent.indy.models.revocation.**IndyRevRegDefValuePublicKeysAccumKeySchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy revocation registry definition value public keys accum key schema.

class Meta

Bases: object

Schema metadata.

model_class
alias of *aries_cloudagent.indy.models.revocation.IndyRevRegDefValuePublicKeysAccumKey*

z

class aries_cloudagent.indy.models.revocation.**IndyRevRegDefValuePublicKeysSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy revocation registry definition value public keys schema.

class Meta
Bases: object

Schema metadata.

model_class
alias of *aries_cloudagent.indy.models.revocation.IndyRevRegDefValuePublicKeys*

accum_key

class aries_cloudagent.indy.models.revocation.**IndyRevRegDefValueSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy revocation registry definition value schema.

class Meta
Bases: object

Schema metadata.

model_class
alias of *aries_cloudagent.indy.models.revocation.IndyRevRegDefValue*

issuance_type

max_cred_num

public_keys

tails_hash

tails_location

class aries_cloudagent.indy.models.revocation.**IndyRevRegEntry**(*ver: Optional[str] = None*, *value: Optional[*aries_cloudagent.indy.models.revocation.IndyR *= None*)

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

Indy revocation registry entry.

class Meta
Bases: object

Model metadata.

schema_class = 'IndyRevRegEntrySchema'

class aries_cloudagent.indy.models.revocation.**IndyRevRegEntrySchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy revocation registry entry schema.

class Meta
Bases: object

Schema metadata.

model_class
alias of *aries_cloudagent.indy.models.revocation.IndyRevRegEntry*

value

ver

class aries_cloudagent.indy.models.revocation.**IndyRevRegEntryValue**(*prev_accum: Optional[str] = None*, *accum: Optional[str] = None*, *revoked: Optional[Sequence[int]] = None*)

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

Indy revocation registry entry value.

class Meta
Bases: object

Model metadata.

schema_class = 'IndyRevRegEntryValueSchema'

class aries_cloudagent.indy.models.revocation.**IndyRevRegEntryValueSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy revocation registry entry value schema.

class Meta
Bases: object

Schema metadata.

model_class = 'IndyRevRegEntryValue'

accum

prev_accum

revoked

aries_cloudagent.indy.models.schema module

Schema artifacts.

class aries_cloudagent.indy.models.schema.**SchemaSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Marshmallow schema for indy schema.

attr_names

ident

name

seqNo

ver

version

aries_cloudagent.indy.models.xform module

aries_cloudagent.indy.sdk package

Submodules

aries_cloudagent.indy.sdk.error module

Indy error handling.

class aries_cloudagent.indy.sdk.error.**IndyErrorHandler**(*message: Optional[str] = None*, *error_cls: Type[aries_cloudagent.core.error.BaseError] = <class 'aries_cloudagent.core.error.BaseError'>*)

Bases: object

Trap IndyError and raise an appropriate LedgerError instead.

classmethod wrap_error(*err_value: indy.error.IndyError*, *message: Optional[str] = None*, *error_cls: Type[aries_cloudagent.core.error.BaseError] = <class 'aries_cloudagent.core.error.BaseError'>*) → *aries_cloudagent.core.error.BaseError*

Create an instance of BaseError from an IndyError.

aries_cloudagent.indy.sdk.holder module

Indy SDK holder implementation.

class aries_cloudagent.indy.sdk.holder.**IndySdkHolder**(*wallet:* aries_cloudagent.indy.sdk.wallet_setup.IndyOpenWallet)

Bases: *aries_cloudagent.indy.holder.IndyHolder*

Indy-SDK holder implementation.

async 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

async create_presentation(*presentation_request: dict*, *requested_credentials: dict*, *schemas: dict*, *credential_definitions: dict*, *rev_states: Optional[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

async 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

async credential_revoked(*ledger:* aries_cloudagent.ledger.base.BaseLedger, *credential_id: str*, *fro: Optional[int] = None*, *to: Optional[int] = None*) → bool

Check ledger for revocation status of credential by cred id.

Parameters **credential_id** – Credential id to check

async delete_credential(*credential_id: str*)

Remove a credential stored in the wallet.

Parameters **credential_id** – Credential id to remove

async get_credential(*credential_id: str*) → str

Get a credential stored in the wallet.

Parameters **credential_id** – Credential id to retrieve

async 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

async 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

async get_mime_type(*credential_id: str*, *attr: Optional[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)

async store_credential(*credential_definition: dict*, *credential_data: dict*, *credential_request_metadata: dict*, *credential_attr_mime_types=None*, *credential_id: Optional[str] = None*, *rev_reg_def: Optional[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

aries_cloudagent.indy.sdk.issuer module

aries_cloudagent.indy.sdk.profile module

aries_cloudagent.indy.sdk.util module

Indy utilities.

async aries_cloudagent.indy.sdk.util.**create_tails_reader**(*tails_file_path: str*) → int
Get a handle for the blob_storage file reader.

async aries_cloudagent.indy.sdk.util.**create_tails_writer**(*tails_base_dir: str*) → int
Get a handle for the blob_storage file writer.

aries_cloudagent.indy.sdk.verifier module

aries_cloudagent.indy.sdk.wallet_plugin module

Utility for loading Postgres wallet plug-in.

aries_cloudagent.indy.sdk.wallet_plugin.**file_ext**()
Determine file extension based on platform.

aries_cloudagent.indy.sdk.wallet_plugin.**load_postgres_plugin**(*storage_config*, *storage_creds*, *raise_exc=False*)
Load postgres dll and configure postgres wallet.

aries_cloudagent.indy.sdk.wallet_setup module

Indy-SDK wallet setup and configuration.

class aries_cloudagent.indy.sdk.wallet_setup.**IndyOpenWallet**(*config:* aries_cloudagent.indy.sdk.wallet_setup.IndyWalletConfig, *created*, *handle*, *master_secret_id: str*)
Bases: object

Handle and metadata for an opened Indy wallet.

async close()
Close previously-opened wallet, removing it if so configured.

property name: str
Accessor for the opened wallet name.

class aries_cloudagent.indy.sdk.wallet_setup.**IndyWalletConfig**(*config: Optional[Mapping[str, Any]] = None*)
Bases: object

A helper class for handling Indy-SDK wallet configuration.

DEFAULT_FRESHNESS = False

DEFAULT_KEY = ''

DEFAULT_KEY_DERIVATION = 'ARGON2I_MOD'

DEFAULT_STORAGE_TYPE = None

KEY_DERIVATION_ARGON2I_INT = 'ARGON2I_INT'

KEY_DERIVATION_ARGON2I_MOD = 'ARGON2I_MOD'

KEY_DERIVATION_RAW = 'RAW'

async create_wallet() → *aries_cloudagent.indy.sdk.wallet_setup.IndyOpenWallet*
Create a new wallet.

Raises

- **ProfileDuplicateError** – If there was an existing wallet with the same name
- **ProfileError** – If there was a problem removing the wallet
- **ProfileError** – If there was another libindy error

async open_wallet(*created: bool = False*) → *aries_cloudagent.indy.sdk.wallet_setup.IndyOpenWallet*
Open wallet, removing and/or creating it if so configured.

Raises

- **ProfileError** – If wallet not found after creation
- **ProfileNotFoundError** – If the wallet is not found
- **ProfileError** – If the wallet is already open
- **ProfileError** – If there is another libindy error

async remove_wallet()
Remove an existing wallet.

Raises

- **ProfileNotFoundError** – If the wallet could not be found
- **ProfileError** – If there was another libindy error

property wallet_access: dict
Accessor the Indy wallet access info.

property wallet_config: dict
Accessor for the Indy wallet config.

Submodules

aries_cloudagent.indy.holder module

Base Indy Holder class.

class aries_cloudagent.indy.holder.**IndyHolder**
Bases: abc.ABC

Base class for holder.

CHUNK = 256

RECORD_TYPE_MIME_TYPES = 'attribute-mime-types'

abstract async 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

abstract async create_presentation(*presentation_request: dict*, *requested_credentials: dict*, *schemas: dict*, *credential_definitions: dict*, *rev_states: Optional[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

abstract async 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

abstract async credential_revoked(*ledger: aries_cloudagent.ledger.base.BaseLedger*, *credential_id: str*, *fro: Optional[int] = None*, *to: Optional[int] = None*) → bool

Check ledger for revocation status of credential by cred id.

Parameters **credential_id** – Credential id to check

abstract async delete_credential(*credential_id: str*)

Remove a credential stored in the wallet.

Parameters **credential_id** – Credential id to remove

abstract async get_credential(*credential_id: str*) → str

Get a credential stored in the wallet.

Parameters **credential_id** – Credential id to retrieve

abstract async get_mime_type(*credential_id: str*, *attr: Optional[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)

abstract async store_credential(*credential_definition: dict*, *credential_data: dict*, *credential_request_metadata: dict*, *credential_attr_mime_types=None*, *credential_id: Optional[str] = None*, *rev_reg_def: Optional[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.indy.holder.**IndyHolderError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base class for holder exceptions.

aries_cloudagent.indy.issuer module

Base Indy Issuer class.

class aries_cloudagent.indy.issuer.**IndyIssuer**

Bases: abc.ABC

Base class for Indy Issuer.

abstract async create_and_store_credential_definition(*origin_did: str*, *schema: dict*, *signature_type: Optional[str] = None*, *tag: Optional[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

abstract async 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*) → 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

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

abstract async create_credential(*schema: dict*, *credential_offer: dict*, *credential_request: dict*, *credential_values: dict*, *revoc_reg_id: Optional[str] = None*, *tails_file_path: Optional[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: The location of the tails file

Returns A tuple of created credential and revocation id

abstract async 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

abstract async create_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

abstract async 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: Optional[str] = None*, *tag: Optional[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.

abstract async 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

abstract async revoke_credentials(*revoc_reg_id: str*, *tails_file_path: str*, *cred_rev_ids: Sequence[str]*) → Tuple[str, Sequence[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_rev_ids** – sequences of credential indexes in the revocation registry

Returns Tuple with the combined revocation delta, list of cred rev ids not revoked

exception aries_cloudagent.indy.issuer.**IndyIssuerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Generic issuer error.

exception aries_cloudagent.indy.issuer.**IndyIssuerRevocationRegistryFullError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.indy.issuer.IndyIssuerError*

Revocation registry is full when issuing a new credential.

aries_cloudagent.indy.util module

Utilities for dealing with Indy conventions.

async aries_cloudagent.indy.util.**generate_pr_nonce**() → str
Generate a nonce for a proof request.

aries_cloudagent.indy.util.**indy_client_dir**(*subpath: Optional[str] = None*, *create: bool = False*) → str
Return '/'-terminated subdirectory of indy-client directory.

Parameters

- **subpath** – subpath within indy-client structure
- **create** – whether to create subdirectory if absent

aries_cloudagent.indy.verifier module

aries_cloudagent.ledger package

Subpackages

aries_cloudagent.ledger.merkel_validation package

Submodules

aries_cloudagent.ledger.merkel_validation.constants module

Constants for State Proof and LeafHash Inclusion Verification.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler module

Utilities for Processing Replies to Domain Read Requests.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**decode_state_value**(*encoded_value*)
Return val, lsn, lut from encoded state value.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**encode_state_value**(*value*, *seqNo*, *txnTime*)
Return encoded state value.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**extract_params_write_request**(*data*)
Return tree_size, leaf_index, audit_path, expected_root_hash from reply.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**get_proof_nodes**(*reply*)
Return proof_nodes from reply.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**hash_of**(*text*) → str
Return 256 bit hexadecimal digest of text.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**make_state_path_for_attr**(*did*, *attr_name*, *attr_is_hash=False*) → bytes
Return state_path for ATTR.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**make_state_path_for_claim_def**(*authors_did*, *schema_seq_no*, *signature_type*, *tag*)
Return state_path for CLAIM DEF.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**make_state_path_for_nym**(*did*) → bytes
Return state_path for NYM.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**make_state_path_for_revoc_def**(*authors_did*, *cred_def_id*, *revoc_def_type*, *revoc_def_tag*) → bytes
Return state_path for REVOC_DEF.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**make_state_path_for_revoc_reg_entry**(*revoc_re* → bytes
Return state_path for REVOC_REG_ENTRY.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**make_state_path_for_revoc_reg_entry_accum**(

Return state_path for REVOC_REG_ENTRY_ACCUM.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**make_state_path_for_schema**(*authors_did*, *schema_name*, *schema_version*) → bytes

Return state_path for SCHEMA.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**parse_attr_txn**(*txn_data*)
Process txn_data and parse attr_txn based on attr_type.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_attr_for_state**(*txn*, *path_only=False*)

Return key, value pair for state from ATTR.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_claim_def_for_state**(*txn*, *path_only=False*)

Return key-value pair for state from CLAIM_DEF.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_for_state_read**(*reply*)
Return state value from read requests reply.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_for_state_write**(*reply*)
Return state key, value pair from write requests reply.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_attr_for_state**(*reply*)
Return value for state from GET_ATTR.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_claim_def_for_state**(*reply*)
Return value for state from GET_CLAIM_DEF.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_nym_for_state**(*reply*)
Return value for state from GET_NYM.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_revoc_def_for_state**(*reply*)
Return value for state from GET_REVOC_DEF.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_revoc_reg_delta_for_state**(*reply*)
Return value for state from GET_REVOC_REG_DELTA.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_revoc_reg_entry_accum_for_stat**
Return value for state from GET_REVOC_REG_ENTRY_ACCUM.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_revoc_reg_entry_for_state**(*reply*)
Return value for state from GET_REVOC_REG_ENTRY.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_get_schema_for_state**(*reply*)
Return value for state from GET_SCHEMA.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_nym_for_state**(*txn*)
Return encoded state path from NYM.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_revoc_def_for_state**(*txn*, *path_only=False*)

Return key-value pair for state from REVOC_DEF.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_revoc_reg_entry_accum_for_state**(*txn*
Return key-value pair for state from REVOC_REG_ENTRY_ACCUM.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_revoc_reg_entry_for_state**(*txn*, *path_only=*

Return key-value pair for state from REVOC_REG_ENTRY.

aries_cloudagent.ledger.merkel_validation.domain_txn_handler.**prepare_schema_for_state**(*txn*, *path_only=False*)

Return key-value pair for state from SCHEMA.

aries_cloudagent.ledger.merkel_validation.hasher module

Merkle tree hasher for leaf and children nodes.

class aries_cloudagent.ledger.merkel_validation.hasher.**HexTreeHasher**(*hashfunc=<built-in function openssl_sha256>*)

Bases: *aries_cloudagent.ledger.merkel_validation.hasher.TreeHasher*

Merkle tree hasher for hex data.

hash_children(*left*, *right*)
Return parent node hash corresponding to 2 child nodes.

hash_leaf(*data*)
Return leaf node hash.

class aries_cloudagent.ledger.merkel_validation.hasher.**TreeHasher**(*hashfunc=<built-in function openssl_sha256>*)

Bases: object

Merkle tree hasher for bytes data.

hash_children(*left*, *right*)
Return parent node hash corresponding to 2 child nodes.

hash_leaf(*data*)
Return leaf node hash.

aries_cloudagent.ledger.merkel_validation.merkel_verifier module

Verify Leaf Inclusion.

class aries_cloudagent.ledger.merkel_validation.merkel_verifier.**MerkleVerifier**(*hasher=<aries_cloudagent.ledg object>*)

Bases: object

Utility class for verifying leaf inclusion.

async calculate_root_hash(*leaf*, *leaf_index*, *audit_path*, *tree_size*)
Calculate root hash, used to verify Merkel AuditPath.

Reference: section 2.1.1 of RFC6962.

Parameters

- **leaf** – Leaf data.
- **leaf_index** – Index of the leaf in the tree.
- **audit_path** – A list of SHA-256 hashes representing the Merkle audit
- **path.** –

- **tree_size** – tree size

lsb(*x*)
Return Least Significant Bits.

aries_cloudagent.ledger.merkel_validation.trie module

Validates State Proof.

class aries_cloudagent.ledger.merkel_validation.trie.**SubTrie**(*root_hash=None*)
Bases: object

Utility class for SubTrie and State Proof validation.

async static get_new_trie_with_proof_nodes(*proof_nodes*)
Return SubTrie created from proof_nodes.

property root_hash
Return 32 bytes string.

set_root_hash(*root_hash=None*)
.

async static verify_spv_proof(*expected_value*, *proof_nodes*, *serialized=True*)
Verify State Proof.

aries_cloudagent.ledger.merkel_validation.utils module

Merkel Validation Utils.

aries_cloudagent.ledger.merkel_validation.utils.**ascii_chr**(*value*)
Return bytes object.

aries_cloudagent.ledger.merkel_validation.utils.**audit_path_length**(*index: int*, *tree_size: int*)
Return AuditPath length.

Parameters

- **index** – Leaf index
- **tree_size** – Tree size

aries_cloudagent.ledger.merkel_validation.utils.**bin_to_nibbles**(*s*)
Convert string s to nibbles (half-bytes).

aries_cloudagent.ledger.merkel_validation.utils.**encode_hex**(*b*)
Return bytes object for string or hexadecimal rep for bytes input.

Parameters **b** – string or bytes

aries_cloudagent.ledger.merkel_validation.utils.**sha3_256**(*x*)
Return 256 bit digest.

aries_cloudagent.ledger.merkel_validation.utils.**unpack_to_nibbles**(*bindata*)
Unpack packed binary data to nibbles.

Parameters **bindata** – binary packed from nibbles

aries_cloudagent.ledger.multiple_ledger package

Submodules

aries_cloudagent.ledger.multiple_ledger.base_manager module

Manager for multiple ledger.

class aries_cloudagent.ledger.multiple_ledger.base_manager.**BaseMultipleLedgerManager**(*profile:* aries_cloudagent.core.pr

Bases: abc.ABC

Base class for handling multiple ledger support.

extract_did_from_identifier(*identifier: str*) → str
Return did from record identifier (REV_REG_ID, CRED_DEF_ID, SCHEMA_ID).

abstract async get_nonprod_ledgers() → Mapping
Return configured non production ledgers.

abstract async get_prod_ledgers() → Mapping
Return configured production ledgers.

abstract async get_write_ledger() → Tuple[str, *aries_cloudagent.ledger.base.BaseLedger*]
Return write ledger.

abstract async lookup_did_in_configured_ledgers(*did: str*, *cache_did: bool*) → Tuple[str, *aries_cloudagent.ledger.base.BaseLedger*]
Lookup given DID in configured ledgers in parallel.

exception aries_cloudagent.ledger.multiple_ledger.base_manager.**MultipleLedgerManagerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Generic multiledger error.

aries_cloudagent.ledger.multiple_ledger.indy_manager module

Multiple IndySdkLedger Manager.

class aries_cloudagent.ledger.multiple_ledger.indy_manager.**MultiIndyLedgerManager**(*profile:* aries_cloudagent.core.profil *production_ledgers:* *collections.OrderedDict* = *{}*, *non_production_ledgers:* *collections.OrderedDict* = *{}*, *write_ledger_info:* *Optional[Tuple[str,* aries_cloudagent.ledger.indy = *None*, *cache_ttl:* *Optional[int]* = *None*)

Bases: *aries_cloudagent.ledger.multiple_ledger.base_manager.BaseMultipleLedgerManager*

Multiple Indy SDK Ledger Manager.

async get_nonprod_ledgers() → Mapping
Return non_production ledgers mapping.

async get_prod_ledgers() → Mapping
Return production ledgers mapping.

async get_write_ledger() → Optional[Tuple[str, *aries_cloudagent.ledger.indy.IndySdkLedger*]]
Return the write IndySdkLedger instance.

async lookup_did_in_configured_ledgers(*did: str*, *cache_did: bool = True*) → Tuple[str, *aries_cloudagent.ledger.indy.IndySdkLedger*]
Lookup given DID in configured ledgers in parallel.

aries_cloudagent.ledger.multiple_ledger.indy_vdr_manager module

Multiple IndyVdrLedger Manager.

class aries_cloudagent.ledger.multiple_ledger.indy_vdr_manager.**MultiIndyVDRLedgerManager**(*profile:* aries_cloudagent.c *pro- duc- tion_ledgers:* *col- lec- tions.OrderedDict* = *{}*, *non_production_le* *col- lec- tions.OrderedDict* = *{}*, *write_ledger_info:* *Op- tional[Tuple[str,* aries_cloudagent.le = *None*, *cache_ttl:* *Op- tional[int]* = *None*)

Bases: *aries_cloudagent.ledger.multiple_ledger.base_manager.BaseMultipleLedgerManager*

Multiple Indy VDR Ledger Manager.

async get_nonprod_ledgers() → Mapping
Return non_production ledgers mapping.

async get_prod_ledgers() → Mapping
Return production ledgers mapping.

async get_write_ledger() → Optional[Tuple[str, *aries_cloudagent.ledger.indy_vdr.IndyVdrLedger*]]
Return the write IndyVdrLedger instance.

async lookup_did_in_configured_ledgers(*did: str*, *cache_did: bool = True*) → Tuple[str, *aries_cloudagent.ledger.indy_vdr.IndyVdrLedger*]
Lookup given DID in configured ledgers in parallel.

aries_cloudagent.ledger.multiple_ledger.ledger_config_schema module

Schema for configuring multiple ledgers.

class aries_cloudagent.ledger.multiple_ledger.ledger_config_schema.**LedgerConfigInstance**(*, *id: Optional[str] = None*, *is_production: str = True*, *genesis_transactions: Optional[str] = None*, *genesis_file: Optional[str] = None*, *genesis_url: Optional[str] = None*)

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

describes each LedgerConfigInstance for multiple ledger support.

class Meta

Bases: object

LedgerConfigInstance metadata.

schema_class = 'LedgerConfigInstanceSchema'

class aries_cloudagent.ledger.multiple_ledger.ledger_config_schema.**LedgerConfigInstanceSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single LedgerConfigInstance Schema.

class Meta

Bases: object

LedgerConfigInstanceSchema metadata.

model_class
alias of *aries_cloudagent.ledger.multiple_ledger.ledger_config_schema.LedgerConfigInstance*

genesis_file

genesis_transactions

genesis_url

id

is_production

validate_id(*data*, ***kwargs*)
Check if id is present, if not then set to UUID4.

class aries_cloudagent.ledger.multiple_ledger.ledger_config_schema.**LedgerConfigListSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for Ledger Config List.

ledger_config_list

class aries_cloudagent.ledger.multiple_ledger.ledger_config_schema.**MultipleLedgerModuleResultSchema**(**arg*
Any,
***k*
Any)

Bases: marshmallow.

Schema for the multiple ledger modules endpoint.

class aries_cloudagent.ledger.multiple_ledger.ledger_config_schema.**WriteLedgerRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for setting/getting ledger_id for the write ledger.

ledger_id

aries_cloudagent.ledger.multiple_ledger.ledger_requests_executor module

Ledger Request Executor.

class aries_cloudagent.ledger.multiple_ledger.ledger_requests_executor.**IndyLedgerRequestsExecutor**(*profile:*
aries_c

Bases: object

Executes Ledger Requests based on multiple ledger config, if set.

async get_ledger_for_identifier(*identifier: str*, *txn_record_type: int*) → Tuple[Optional[str], Optional[*aries_cloudagent.ledger.base.BaseLedger*]]
Return ledger info given the record identifier.

aries_cloudagent.ledger.multiple_ledger.manager_provider module

Profile manager for multiple Indy ledger support.

class aries_cloudagent.ledger.multiple_ledger.manager_provider.**MultiIndyLedgerManagerProvider**(*root_profile*)

Bases: *aries_cloudagent.config.base.BaseProvider*

Multiple Indy ledger support manager provider.

LEDGER_TYPES = {'askar-profile': {'ledger': <aries_cloudagent.utils.classloader.DeferLoad object>, 'pool': <aries_cloudagent.utils.classloader.DeferLoad object>}, 'basic': {'ledger': <aries_cloudagent.utils.classloader.DeferLoad object>, 'pool': <aries_cloudagent.utils.classloader.DeferLoad object>}}

MANAGER_TYPES = {'askar-profile': <aries_cloudagent.utils.classloader.DeferLoad object>, 'basic': <aries_cloudagent.utils.classloader.DeferLoad object>}

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

Create the multiple Indy ledger manager instance.

Submodules

aries_cloudagent.ledger.base module

Ledger base class.

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

Bases: abc.ABC

Base class for ledger.

BACKEND_NAME: str = None

abstract async accept_txn_author_agreement(*taa_record: dict*, *mechanism: str*, *accept_time: Optional[int] = None*)

Save a new record recording the acceptance of the TAA.

property backend: str

Accessor for the ledger backend name.

async 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.

async create_and_send_credential_definition(*issuer:* aries_cloudagent.indy.issuer.IndyIssuer, *schema_id: str*, *signature_type: Optional[str] = None*, *tag: Optional[str] = None*, *support_revocation: bool = False*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → Tuple[str, dict, bool]

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

Returns Tuple with cred def id, cred def structure, and whether it's novel

async create_and_send_schema(*issuer:* aries_cloudagent.indy.issuer.IndyIssuer, *schema_name: str*, *schema_version: str*, *attribute_names: Sequence[str]*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → 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.

abstract async 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

abstract async fetch_schema_by_seq_no(*seq_no: int*) → dict

Fetch a schema by its sequence number.

Parameters **seq_no** – schema ledger sequence number

Returns Indy schema dict

abstract async fetch_txn_author_agreement()

Fetch the current AML and TAA from the ledger.

abstract async get_all_endpoints_for_did(*did: str*) → dict

Fetch all endpoints for a ledger DID.

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

abstract async 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

abstract async get_endpoint_for_did(*did: str*, *endpoint_type:* aries_cloudagent.ledger.endpoint_type.EndpointType = *EndpointType.ENDPOINT*) → str

Fetch the endpoint for a ledger DID.

Parameters

- **did** – The DID to look up on the ledger or in the cache
- **endpoint_type** – The type of the endpoint (default 'endpoint')

abstract async 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

abstract async get_latest_txn_author_acceptance()
Look up the latest TAA acceptance.

abstract async get_nym_role(*did: str*)
Return the role registered to input public DID on the ledger.

Parameters **did** – DID to register on the ledger.

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

abstract async get_revoc_reg_delta(*revoc_reg_id: str*, *timestamp_from=0*, *timestamp_to=None*) → Tuple[dict, int]
Look up a revocation registry delta by ID.

abstract async get_revoc_reg_entry(*revoc_reg_id: str*, *timestamp: int*) → Tuple[dict, int]
Get revocation registry entry by revocation registry ID and timestamp.

abstract async 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

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

abstract async get_wallet_public_did() → *aries_cloudagent.wallet.did_info.DIDInfo*
Fetch the public DID from the wallet.

abstract async is_ledger_read_only() → bool
Check if ledger is read-only including TAA.

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

abstract property read_only: bool
Accessor for the ledger read-only flag.

abstract async register_nym(*did: str*, *verkey: str*, *alias: Optional[str] = None*, *role: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → Tuple[bool, dict]
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.

abstract async rotate_public_did_keypair(*next_seed: Optional[str] = None*) → None
Rotate keypair for public DID: create new key, submit to ledger, update wallet.

Parameters **next_seed** – seed for incoming ed25519 keypair (default random)

abstract async send_revoc_reg_def(*revoc_reg_def: dict*, *issuer_did: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict
Publish a revocation registry definition to the ledger.

abstract async send_revoc_reg_entry(*revoc_reg_id: str*, *revoc_def_type: str*, *revoc_reg_entry: dict*, *issuer_did: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict

Publish a revocation registry entry to the ledger.

taa_digest(*version: str*, *text: str*)

Generate the digest of a TAA record.

abstract async txn_endorse(*request_json: str*, *endorse_did: Optional[*aries_cloudagent.wallet.did_info.DIDInfo*] = None*) → str

Endorse (sign) the provided transaction.

abstract async txn_submit(*request_json: str*, *sign: bool*, *taa_accept: Optional[bool] = None*, *sign_did: aries_cloudagent.wallet.did_info.DIDInfo = <object object>*, *write_ledger: bool = True*) → str

Write the provided (signed and possibly endorsed) transaction to the ledger.

abstract async update_endpoint_for_did(*did: str*, *endpoint: str*, *endpoint_type:* aries_cloudagent.ledger.endpoint_type.EndpointType *= EndpointType.ENDPOINT*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*, *routing_keys: Optional[List[str]] = None*) → bool

Check and update the endpoint on the ledger.

Parameters

- **did** – The ledger DID
- **endpoint** – The endpoint address
- **endpoint_type** – The type of the endpoint (default 'endpoint')

class aries_cloudagent.ledger.base.**Role**(*value*)

Bases: enum.Enum

Enum for indy roles.

ENDORSER = (101,)

NETWORK_MONITOR = (201,)

ROLE_REMOVE = ('',)

STEWARD = (2,)

TRUSTEE = (0,)

USER = (None, '')

static get(*token: Optional[Union[str, int]] = None*) → *aries_cloudagent.ledger.base.Role*

Return enum instance corresponding to input token.

Parameters token – token identifying role to indy-sdk: "STEWARD", "TRUSTEE", "ENDORSER", "" or None

to_indy_num_str() → str

Return (typically, numeric) string value that indy-sdk associates with role.

Recall that None signifies USER and "" signifies a role undergoing reset.

token() → str

Return token identifying role to indy-sdk.

aries_cloudagent.ledger.endpoint_type module

Ledger utilities.

class aries_cloudagent.ledger.endpoint_type.**EndpointType**(*value*)

Bases: enum.Enum

Enum for endpoint/service types.

ENDPOINT = EndpointTypeName(w3c='Endpoint', indy='endpoint')

LINKED_DOMAINS = EndpointTypeName(w3c='LinkedDomains', indy='linked_domains')

PROFILE = EndpointTypeName(w3c='Profile', indy='profile')

static get(*name: str*) → *aries_cloudagent.ledger.endpoint_type.EndpointType*

Return enum instance corresponding to input string.

property indy

internally-facing, on ledger and in wallet.

Type Indy name of endpoint type

property w3c

externally-facing.

Type W3C name of endpoint type

class aries_cloudagent.ledger.endpoint_type.**EndpointTypeName**(*w3c*, *indy*)

Bases: tuple

property indy

Alias for field number 1

property w3c

Alias for field number 0

aries_cloudagent.ledger.error module

Ledger related errors.

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

Bases: *aries_cloudagent.ledger.error.LedgerError*

The current request cannot proceed.

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

Bases: *aries_cloudagent.ledger.error.LedgerError*

Indy pool is closed.

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

Bases: *aries_cloudagent.ledger.error.LedgerError*

Base class for ledger configuration errors.

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

Bases: *aries_cloudagent.core.error.BaseError*

Base class for ledger errors.

exception aries_cloudagent.ledger.error.**LedgerTransactionError**(**args*, *error_code: Optional[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.**IndySdkLedger**(*pool: aries_cloudagent.ledger.indy.IndySdkLedgerPool*, *profile: IndySdkProfile*)

Bases: *aries_cloudagent.ledger.base.BaseLedger*

Indy ledger class.

BACKEND_NAME: str = 'indy'

async **accept_txn_author_agreement**(*taa_record: dict*, *mechanism: str*, *accept_time: Optional[int] = None*)

Save a new record recording the acceptance of the TAA.

async **build_and_return_get_nym_request**(*submitter_did: Optional[str]*, *target_did: str*) → str

Build GET_NYM request and return request_json.

async **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

async **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

async **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

async **fetch_schema_by_seq_no**(*seq_no: int*) → dict

Fetch a schema by its sequence number.

Parameters **seq_no** – schema ledger sequence number

Returns Indy schema dict

async **fetch_txn_author_agreement**() → dict

Fetch the current AML and TAA from the ledger.

async **get_all_endpoints_for_did**(*did: str*) → dict

Fetch all endpoints for a ledger DID.

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

async **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

async get_endpoint_for_did(*did: str*, *endpoint_type: Optional[aries_cloudagent.ledger.endpoint_type.EndpointType] = None*) → str

Fetch the endpoint for a ledger DID.

Parameters

- **did** – The DID to look up on the ledger or in the cache
- **endpoint_type** – The type of the endpoint. If none given, returns all

async get_indy_storage() → *aries_cloudagent.storage.indy.IndySdkStorage*

Get an IndySdkStorage instance for the current wallet.

async 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

async get_latest_txn_author_acceptance() → dict

Look up the latest TAA acceptance.

async get_nym_role(*did: str*) → *aries_cloudagent.ledger.base.Role*

Return the role of the input public DID's NYM on the ledger.

Parameters **did** – DID to query for role on the ledger.

async get_revoc_reg_def(*revoc_reg_id: str*) → dict

Get revocation registry definition by ID; augment with ledger timestamp.

async get_revoc_reg_delta(*revoc_reg_id: str*, *fro=0*, *to=None*) → Tuple[dict, int]

Look up a revocation registry delta by ID.

:param revoc_reg_id revocation registry id :param fro earliest EPOCH time of interest :param to latest EPOCH time of interest

:returns delta response, delta timestamp

async get_revoc_reg_entry(*revoc_reg_id: str*, *timestamp: int*)

Get revocation registry entry by revocation registry ID and timestamp.

async 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

async get_txn_author_agreement(*reload: bool = False*) → dict

Get the current transaction author agreement, fetching it if necessary.

async get_wallet_public_did() → *aries_cloudagent.wallet.did_info.DIDInfo*

Fetch the public DID from the wallet.

async is_ledger_read_only() → bool

Check if ledger is read-only including TAA.

nym_to_did(*nym: str*) → str

Format a nym with the ledger's DID prefix.

property pool_handle

Accessor for the ledger pool handle.

property pool_name: str

Accessor for the ledger pool name.

property read_only: bool
Accessor for the ledger read-only flag.

async register_nym(*did: str*, *verkey: str*, *alias: Optional[str] = None*, *role: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → Tuple[bool, dict]
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.

async rotate_public_did_keypair(*next_seed: Optional[str] = None*) → None
Rotate keypair for public DID: create new key, submit to ledger, update wallet.

Parameters **next_seed** – seed for incoming ed25519 keypair (default random)

async send_revoc_reg_def(*revoc_reg_def: dict*, *issuer_did: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict
Publish a revocation registry definition to the ledger.

async send_revoc_reg_entry(*revoc_reg_id: str*, *revoc_def_type: str*, *revoc_reg_entry: dict*, *issuer_did: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict
Publish a revocation registry entry to the ledger.

async submit_get_nym_request(*request_json: str*) → str
Submit GET_NYM request to ledger and return response_json.

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

Anything more accurate is a privacy concern.

async txn_endorse(*request_json: str*, *endorse_did: Optional[aries_cloudagent.wallet.did_info.DIDInfo] = None*) → str
Endorse a (signed) ledger transaction.

async txn_submit(*request_json: str*, *sign: Optional[bool] = None*, *taa_accept: Optional[bool] = None*, *sign_did: aries_cloudagent.wallet.did_info.DIDInfo = <object object>*, *write_ledger: bool = True*) → str
Submit a signed (and endorsed) transaction to the ledger.

async update_endpoint_for_did(*did: str*, *endpoint: str*, *endpoint_type: Optional[aries_cloudagent.ledger.endpoint_type.EndpointType] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*, *routing_keys: Optional[List[str]] = None*) → bool
Check and update the endpoint on the ledger.

Parameters

- **did** – The ledger DID
- **endpoint** – The endpoint address
- **endpoint_type** – The type of the endpoint

class aries_cloudagent.ledger.indy.**IndySdkLedgerPool**(*name: str*, *, *checked: bool = False*, *keepalive: int = 0*, *cache: Optional[aries_cloudagent.cache.base.BaseCache] = None*, *cache_duration: int = 600*, *genesis_transactions: Optional[str] = None*, *read_only: bool = False*, *socks_proxy: Optional[str] = None*)

Bases: object

Indy ledger manager class.

async check_pool_config() → bool
Check if a pool config has been created.

async close()
Close the pool ledger.

async context_close()
Release the reference and schedule closing of the pool ledger.

async context_open()
Open the ledger if necessary and increase the number of active references.

async create_pool_config(*genesis_transactions: str*, *recreate: bool = False*)
Create the pool ledger configuration.

property genesis_txns: str
Get the configured genesis transactions.

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

class aries_cloudagent.ledger.indy.**IndySdkLedgerPoolProvider**

Bases: *aries_cloudagent.config.base.BaseProvider*

Indy ledger pool provider which keys off the selected pool name.

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

aries_cloudagent.ledger.indy_vdr module

Indy-VDR ledger implementation.

class aries_cloudagent.ledger.indy_vdr.**IndyVdrLedger**(*pool:* aries_cloudagent.ledger.indy_vdr.IndyVdrLedgerPool, *profile:* aries_cloudagent.core.profile.Profile)

Bases: *aries_cloudagent.ledger.base.BaseLedger*

Indy-VDR ledger class.

BACKEND_NAME: str = 'indy-vdr'

async accept_txn_author_agreement(*taa_record: dict*, *mechanism: str*, *accept_time: Optional[int] = None*)
Save a new record recording the acceptance of the TAA.

async build_and_return_get_nym_request(*submitter_did: Optional[str]*, *target_did: str*) → str
Build GET_NYM request and return request_json.

async 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

async 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

async 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

async fetch_schema_by_seq_no(*seq_no: int*) → dict
Fetch a schema by its sequence number.

Parameters **seq_no** – schema ledger sequence number

Returns Indy schema dict

async fetch_txn_author_agreement() → dict
Fetch the current AML and TAA from the ledger.

async get_all_endpoints_for_did(*did: str*) → dict
Fetch all endpoints for a ledger DID.

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

async 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

async get_endpoint_for_did(*did: str*, *endpoint_type: Optional[*aries_cloudagent.ledger.endpoint_type.EndpointType*] = None*) → str
Fetch the endpoint for a ledger DID.

Parameters

- **did** – The DID to look up on the ledger or in the cache
- **endpoint_type** – The type of the endpoint. If none given, returns all

async 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

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

async get_nym_role(*did: str*) → *aries_cloudagent.ledger.base.Role*
Return the role of the input public DID's NYM on the ledger.

Parameters **did** – DID to query for role on the ledger.

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

async get_revoc_reg_delta(*revoc_reg_id: str*, *timestamp_from=0*, *timestamp_to=None*) → Tuple[dict, 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

async get_revoc_reg_entry(*revoc_reg_id: str*, *timestamp: int*) → Tuple[dict, int]
Get revocation registry entry by revocation registry ID and timestamp.

async 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

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

async get_wallet_public_did() → *aries_cloudagent.wallet.did_info.DIDInfo*
Fetch the public DID from the wallet.

async is_ledger_read_only() → bool
Check if ledger is read-only including TAA.

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

property pool_handle
Accessor for the ledger pool handle.

property pool_name: str
Accessor for the ledger pool name.

property read_only: bool
Accessor for the ledger read-only flag.

async register_nym(*did: str*, *verkey: str*, *alias: Optional[str] = None*, *role: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → Tuple[bool, dict]
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.

async rotate_public_did_keypair(*next_seed: Optional[str] = None*) → None
Rotate keypair for public DID: create new key, submit to ledger, update wallet.

Parameters **next_seed** – seed for incoming ed25519 keypair (default random)

async send_revoc_reg_def(*revoc_reg_def: dict*, *issuer_did: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict
Publish a revocation registry definition to the ledger.

async send_revoc_reg_entry(*revoc_reg_id: str*, *revoc_def_type: str*, *revoc_reg_entry: dict*, *issuer_did: Optional[str] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict
Publish a revocation registry entry to the ledger.

async **submit_get_nym_request**(*request_json: str*) → str
Submit GET_NYM request to ledger and return response_json.

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

Anything more accurate is a privacy concern.

async **txn_endorse**(*request_json: str*, *endorse_did: Optional[*aries_cloudagent.wallet.did_info.DIDInfo*] = None*) → str
Endorse (sign) the provided transaction.

async **txn_submit**(*request_json: str*, *sign: bool*, *taa_accept: Optional[bool] = None*, *sign_did: aries_cloudagent.wallet.did_info.DIDInfo = <object object>*, *write_ledger: bool = True*) → str
Write the provided (signed and possibly endorsed) transaction to the ledger.

async **update_endpoint_for_did**(*did: str*, *endpoint: str*, *endpoint_type: Optional[*aries_cloudagent.ledger.endpoint_type.EndpointType*] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*, *routing_keys: Optional[List[str]] = None*) → bool
Check and update the endpoint on the ledger.

Parameters

- **did** – The ledger DID
- **endpoint** – The endpoint address
- **endpoint_type** – The type of the endpoint

class aries_cloudagent.ledger.indy_vdr.**IndyVdrLedgerPool**(*name: str*, ***, *keepalive: int = 0*, *cache: Optional[*aries_cloudagent.cache.base.BaseCache*] = None*, *cache_duration: int = 600*, *genesis_transactions: Optional[str] = None*, *read_only: bool = False*, *socks_proxy: Optional[str] = None*)

Bases: object

Indy-VDR ledger pool manager.

property **cfg_path**: pathlib.Path
Get the path to the configuration file, ensuring it's created.

async **close**()
Close the pool ledger.

async **context_close**()
Release the reference and schedule closing of the pool ledger.

async **context_open**()
Open the ledger if necessary and increase the number of active references.

async **create_pool_config**(*genesis_transactions: str*, *recreate: bool = False*)
Create the pool ledger configuration.

property **genesis_hash**: str
Get the hash of the configured genesis transactions.

property **genesis_txns**: str
Get the configured genesis transactions.

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

aries_cloudagent.ledger.routes module

aries_cloudagent.ledger.util module

Ledger utilities.

async aries_cloudagent.ledger.util.**notify_register_did_event**(*profile:* aries_cloudagent.core.profile.Profile, *did: str*, *meta_data: dict*)

Send notification for a DID post-process event.

aries_cloudagent.messaging package

Subpackages

aries_cloudagent.messaging.credential_definitions package

Submodules

aries_cloudagent.messaging.credential_definitions.routes module

aries_cloudagent.messaging.credential_definitions.util module

Credential definition utilities.

class aries_cloudagent.messaging.credential_definitions.util.**CredDefQueryStringSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query string parameters for credential definition searches.

cred_def_id

issuer_did

schema_id

schema_issuer_did

schema_name

schema_version

async aries_cloudagent.messaging.credential_definitions.util.**notify_cred_def_event**(*profile:* aries_cloudagent.core.prof *cred_def_id: str*, *meta_data: dict*)

Send notification for a cred def post-process event.

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: Optional[str] = None*, *description: Optional[str] = None*, *filename: Optional[str] = None*, *mime_type: Optional[str] = None*, *lastmod_time: Optional[str] = None*, *byte_count: Optional[int] = None*, *data: aries_cloudagent.messaging.decor*, ***kwargs*)

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

Class representing attach decorator.

class Meta

Bases: object

AttachDecorator metadata.

schema_class = 'AttachDecoratorSchema'

property content: Union[Mapping, Tuple[Sequence[str], str]]

Return attachment content.

Returns data attachment, decoded if necessary and json-loaded, or data links and sha-256 hash.

classmethod data_base64(*mapping: Mapping*, ***, *ident: Optional[str] = None*, *description: Optional[str] = None*, *filename: Optional[str] = None*, *lastmod_time: Optional[str] = None*, *byte_count: Optional[int] = None*)

Create *AttachDecorator* instance on base64-encoded data from input mapping.

Given mapping, JSON dump, base64-encode, and embed it as data; mark *application/json* MIME type.

Parameters

- **mapping** – (dict) data structure; e.g., indy production
- **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

classmethod data_json(*mapping: Union[Sequence[dict], dict]*, ***, *ident: Optional[str] = None*, *description: Optional[str] = None*, *filename: Optional[str] = None*, *lastmod_time: Optional[str] = None*, *byte_count: Optional[int] = None*)

Create *AttachDecorator* instance on json-encoded data from input mapping.

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

Parameters

- **mapping** – (dict) data structure; e.g., Aries message
- **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

classmethod data_links(*links: Union[str, Sequence[str]]*, *sha256: Optional[str] = None*, ***, *ident: Optional[str] = None*, *mime_type: Optional[str] = None*, *description: Optional[str] = None*, *filename: Optional[str] = None*, *lastmod_time: Optional[str] = None*, *byte_count: Optional[int] = None*)

Create *AttachDecorator* instance on json-encoded data from input mapping.

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

Parameters

- **links** – URL or list of URLs
- **sha256** – optional sha-256 hash for content
- **ident** – optional attachment identifier (default random UUID4)
- **mime_type** – optional MIME type
- **description** – optional attachment description
- **filename** – optional attachment filename
- **lastmod_time** – optional attachment last modification time
- **byte_count** – optional attachment byte count

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorData**(*, *jws_: Optional[aries_cloudagent.mess = None*, *sha256_: Optional[str] = None*, *links_: Optional[Union[Sequence[str], str]] = None*, *base64_: Optional[str] = None*, *json_: Optional[Union[Sequence[dict], dict]] = None*)

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

Attach decorator data.

class Meta

Bases: object

AttachDecoratorData metadata.

schema_class = 'AttachDecoratorDataSchema'

property base64

Accessor for base64 decorator data, or None.

header_map(*idx: int = 0*, *jose: bool = True*) → Mapping

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

property json

Accessor for json decorator data, or None.

property jws

Accessor for JWS, or None.

property links

Accessor for links decorator data, or None.

property sha256

Accessor for sha256 decorator data, or None.

async 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

property signatures: int
Accessor for number of signatures.

property signed: bytes
Accessor for signed content (payload), None for unsigned.

async verify(*wallet:* aries_cloudagent.wallet.base.BaseWallet, *signer_verkey: Optional[str] = None*) → 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_cloudagent.messa *protected: Optional[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**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single attach decorator data JWS schema.

class Meta
Bases: object

Single attach decorator data JWS schema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorData1JWS*

header
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorDataJWSHeaderSchema*

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataJWS**(*, *header: Optional[aries_cloudagent.m* = *None*, *protected: Optional[str]* = *None*, *signature: Optional[str]* = *None*, *signatures: Optional[Sequence[aries_cl* = *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**(*kid: str*)

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.**AttachDecoratorDataJWSHeaderSchema**(**args: Any*, ***kwargs Any*)

Bases: marshmallow.

Attach decorator data JWS header schema.

class Meta

Bases: object

Attach decorator data schema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorDataJWSHeader*

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataJWSSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

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 *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorDataJWS*

header
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorDataJWSHeaderSchema*

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

class aries_cloudagent.messaging.decorators.attach_decorator.**AttachDecoratorDataSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Attach decorator data schema.

class Meta
Bases: object

Attach decorator data schema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorData*

json_
Dict or Dict List field for Marshmallow.

jws_
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorDataJWSSchema*

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

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

Bases: marshmallow.

Attach decorator schema used in serialization/deserialization.

class Meta
Bases: object

AttachDecoratorSchema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*

data
alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorDataSchema*

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: Optional[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*, *schema: Optional[Type[marshmallow.Schema]] = None*, *serialized: bool = True*, *skip_attrs: Optional[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.

property fields: collections.OrderedDict
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.

property models: dict
Accessor for the models dictionary.

property prefix: str
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: Optional[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: Optional[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: Optional[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**(**, locale: Optional[str] = None, localizable: Optional[Sequence[st = None, catalogs: Optional[Sequence[st = None*)

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.LocalizationDecoratorSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Localization decorator schema used in serialization/deserialization.

class Meta
Bases: object

LocalizationDecoratorSchema metadata.

model_class
alias of aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecorator

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_id: Optional[str] = None, on: Optional[Sequence[str]] = 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(*args: Any, **kwargs: Any)

Bases: marshmallow.

PleaseAck decorator schema used in serialization/deserialization.

class Meta
Bases: object

PleaseAckDecoratorSchema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.please_ack_decorator.PleaseAckDecorator*

message_id

on

aries_cloudagent.messaging.decorators.service_decorator module

A message decorator for services.

A service decorator adds routing information to a message so agent can respond without needing to perform a handshake.

class aries_cloudagent.messaging.decorators.service_decorator.**ServiceDecorator**(*, *endpoint: str*, *recipient_keys: List[str]*, *routing_keys: Optional[List[str]] = None*)

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

Class representing service decorator.

class Meta
Bases: object

ServiceDecorator metadata.

schema_class = 'ServiceDecoratorSchema'

property endpoint
Accessor for service endpoint.

Returns This service's *serviceEndpoint*

property recipient_keys
Accessor for recipient keys.

Returns This service's *recipientKeys*

property routing_keys
Accessor for routing keys.

Returns This service's *routingKeys*

class aries_cloudagent.messaging.decorators.service_decorator.**ServiceDecoratorSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Thread decorator schema used in serialization/deserialization.

class Meta
Bases: object

ServiceDecoratorSchema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.service_decorator.ServiceDecorator*

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: Optional[str] = None*, *signature: Optional[str] = None*, *sig_data: Optional[str] = None*, *signer: Optional[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 = 'signature/1.0/ed25519Sha512_single'

async 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)

async 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**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

SignatureDecorator schema.

class Meta
Bases: object

SignatureDecoratorSchema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator*

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: Optional[str] = None*, *pthid: Optional[str] = None*, *sender_order: Optional[int] = None*, *received_orders: Optional[Mapping] = None*)

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

Class representing thread decorator.

class Meta
Bases: object

ThreadDecorator metadata.

schema_class = 'ThreadDecoratorSchema'

property pthid
Accessor for parent thread identifier.

Returns This thread's *pthid*

property received_orders: dict
Get received orders.

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

property sender_order: int
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

property thid
Accessor for thread identifier.

Returns This thread's *thid*

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

Bases: marshmallow.

Thread decorator schema used in serialization/deserialization.

class Meta
Bases: object

ThreadDecoratorSchema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.thread_decorator.ThreadDecorator*

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: Optional[Union[str, datetime.datetime]] = None*, *out_time: Optional[Union[str, datetime.datetime]] = None*, *stale_time: Optional[Union[str, datetime.datetime]] = None*, *expires_time: Optional[Union[str, datetime.datetime]] = None*, *delay_milli: Optional[int] = None*, *wait_until_time: Optional[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: Any*, ***kwargs: Any*)

Bases: marshmallow.

Timing decorator schema used in serialization/deserialization.

class Meta

Bases: object

TimingDecoratorSchema metadata.

model_class

alias of *aries_cloudagent.messaging.decorators.timing_decorator.TimingDecorator*

aries_cloudagent.messaging.decorators.trace_decorator module

A message decorator for trace events.

A trace decorator identifies a responsibility on the processor to record information on message processing events.

class aries_cloudagent.messaging.decorators.trace_decorator.**TraceDecorator**(*, *target: Optional[str] = None*, *full_thread: bool = True*, *trace_reports: Optional[Sequence] = None*)

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

Class representing trace decorator.

class Meta

Bases: object

TraceDecorator metadata.

schema_class = 'TraceDecoratorSchema'

append_trace_report(*trace_report:* aries_cloudagent.messaging.decorators.trace_decorator.TraceReport)

Append a trace report to this decorator.

property full_thread

Accessor for full_thread flag.

Returns The full_thread flag

property target

Accessor for trace target.

Returns The target for tracing messages

property trace_reports

Set of trace reports for this message.

Returns The trace reports that have been logged on this message/thread so far. (Only for target="message".)

class aries_cloudagent.messaging.decorators.trace_decorator.**TraceDecoratorSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Trace decorator schema used in serialization/deserialization.

class Meta

Bases: object

TraceDecoratorSchema metadata.

model_class

alias of *aries_cloudagent.messaging.decorators.trace_decorator.TraceDecorator*

class aries_cloudagent.messaging.decorators.trace_decorator.**TraceReport**(*, *msg_id: Optional[str] = None*, *thread_id: Optional[str] = None*, *traced_type: Optional[str] = None*, *timestamp: Optional[str] = None*, *str_time: Optional[str] = None*, *handler: Optional[str] = None*, *ellapsed_milli: Optional[int] = None*, *outcome: Optional[str] = None*)

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

Class representing a Trace Report.

class Meta

Bases: object

TraceReport metadata.

schema_class = 'TraceReport'

property ellapsed_milli

Accessor for ellapsed_milli.

Returns The sender ellapsed_milli

property handler

Accessor for handler.

Returns The sender handler

property msg_id

Accessor for msg_id.

Returns The msg_id

property outcome

Accessor for outcome.

Returns The sender outcome

property str_time

Accessor for str_time.

Returns Formatted representation of the sender timestamp

property thread_id

Accessor for thread_id.

Returns The thread_id

property timestamp

Accessor for timestamp.

Returns The sender timestamp

property traced_type
Accessor for traced_type.

Returns The sender traced_type

class aries_cloudagent.messaging.decorators.trace_decorator.**TraceReportSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Trace report schema.

class Meta
Bases: object

TraceReportSchema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.trace_decorator.TraceReport*

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: Optional[str] = None, return_route_thread: Optional[str] = None, queued_message_count: Optional[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**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Transport decorator schema used in serialization/deserialization.

class Meta
Bases: object

TransportDecoratorSchema metadata.

model_class
alias of *aries_cloudagent.messaging.decorators.transport_decorator.TransportDecorator*

aries_cloudagent.messaging.jsonld package

Submodules

aries_cloudagent.messaging.jsonld.create_verify_data module

Contains the functions needed to produce and verify a json-ld signature.

This file was ported from https://github.com/transmute-industries/Ed25519Signature2018/blob/master/src/createVerifyData/index.js

aries_cloudagent.messaging.jsonld.create_verify_data.**create_verify_data**(*data*, *signature_options*, *document_loader=None*)
Encapsulate process of constructing string used during sign and verify.

aries_cloudagent.messaging.jsonld.credential module

Sign and verify functions for json-ld based credentials.

aries_cloudagent.messaging.jsonld.credential.**b64decode**(*bytes*)
Url Safe B64 Decode.

aries_cloudagent.messaging.jsonld.credential.**b64encode**(*str*)
Url Safe B64 Encode.

aries_cloudagent.messaging.jsonld.credential.**create_jws**(*encoded_header*, *verify_data*)
Compose JWS.

aries_cloudagent.messaging.jsonld.credential.**did_key**(*verkey: str*) → str
Qualify verkey into DID key if need be.

async aries_cloudagent.messaging.jsonld.credential.**jws_sign**(*session*, *verify_data*, *verkey*)
Sign JWS.

async aries_cloudagent.messaging.jsonld.credential.**jws_verify**(*session*, *verify_data*, *signature*, *public_key*)
Detatched jws verify handling.

async aries_cloudagent.messaging.jsonld.credential.**sign_credential**(*session*, *credential*, *signature_options*, *verkey*)
Sign Credential.

async aries_cloudagent.messaging.jsonld.credential.**verify_credential**(*session*, *doc*, *verkey*)
Verify credential.

aries_cloudagent.messaging.jsonld.credential.**verify_jws_header**(*header*)
Check header requirements.

aries_cloudagent.messaging.jsonld.error module

JSON-LD messaging Exceptions.

exception aries_cloudagent.messaging.jsonld.error.**BadJWSHeaderError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.jsonld.error.BaseJSONLDMessagingError*

Exception indicating invalid JWS header.

exception aries_cloudagent.messaging.jsonld.error.**BaseJSONLDMessagingError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base exception class for JSON-LD messaging.

exception aries_cloudagent.messaging.jsonld.error.**DroppedAttributeError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.jsonld.error.BaseJSONLDMessagingError*

Exception used to track that an attribute was removed.

exception aries_cloudagent.messaging.jsonld.error.**InvalidVerificationMethod**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.jsonld.error.BaseJSONLDMessagingError*

Exception indicating an invalid verification method in doc to verify.

exception aries_cloudagent.messaging.jsonld.error.**MissingVerificationMethodError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.jsonld.error.BaseJSONLDMessagingError*

Exception indicating missing verification method from signature options.

exception aries_cloudagent.messaging.jsonld.error.**SignatureTypeError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.jsonld.error.BaseJSONLDMessagingError*

Exception indicating Signature type error.

aries_cloudagent.messaging.jsonld.routes module

jsonld admin routes.

class aries_cloudagent.messaging.jsonld.routes.**DocSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Schema for LD doc to sign.

credential

options

class aries_cloudagent.messaging.jsonld.routes.**SignRequestSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Request schema for signing a jsonld doc.

doc

verkey

class aries_cloudagent.messaging.jsonld.routes.**SignResponseSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Response schema for a signed jsonld doc.

error

signed_doc

class aries_cloudagent.messaging.jsonld.routes.**SignatureOptionsSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Schema for LD signature options.

challenge

domain

proofPurpose

type

verificationMethod

class aries_cloudagent.messaging.jsonld.routes.**SignedDocSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Verifiable doc schema.

class Meta
Bases: object

Keep unknown values.

proof

class aries_cloudagent.messaging.jsonld.routes.**VerifyRequestSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Request schema for signing a jsonld doc.

doc

verkey

class aries_cloudagent.messaging.jsonld.routes.**VerifyResponseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Response schema for verification result.

error

valid

aries_cloudagent.messaging.jsonld.routes.**post_process_routes**(*app: aiohttp.web.Application*)
Amend swagger API.

async aries_cloudagent.messaging.jsonld.routes.**register**(*app: aiohttp.web.Application*)
Register routes.

aries_cloudagent.messaging.models package

Common code for messaging models.

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

property **Schema: Type**[*aries_cloudagent.messaging.models.base.BaseModelSchema*]
Accessor for the model's schema class.

Returns The schema class

classmethod **deserialize**(*obj*, ***, *unknown: Optional[str] = 'None'*) → aries_cloudagent.messaging.models.base.ModelType
classmethod **deserialize**(*obj*, ***, *none2none: typing_extensions.Literal.False*, *unknown: Optional[str] = 'None'*) → aries_cloudagent.messaging.models.base.ModelType
classmethod **deserialize**(*obj*, ***, *none2none: typing_extensions.Literal.True*, *unknown: Optional[str] = 'None'*) → Optional[aries_cloudagent.messaging.models.base.ModelType]
Convert from JSON representation to a model instance.

Parameters

- **obj** – The dict to load into a model instance
- **unknown** – Behaviour for unknown attributes
- **none2none** – Deserialize None to None

Returns A model instance for this data

classmethod from_json(*json_repr: Union[str, bytes]*, *unknown: Optional[str] = None*)
Parse a JSON string into a model instance.

Parameters **json_repr** – JSON string

Returns A model instance representation of this JSON

classmethod serde(*obj: Union[*aries_cloudagent.messaging.models.base.BaseModel*, Mapping]*) → Optional[*aries_cloudagent.messaging.models.base.SerDe*]
Return serialized, deserialized representations of input object.

serialize(*, *as_string: typing_extensions.Literal.True*, *unknown: Optional[str] = 'None'*) → str
serialize(*, *unknown: Optional[str] = 'None'*) → 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(*unknown: Optional[str] = None*) → str
Create a JSON representation of the model instance.

Returns A JSON representation of this message

validate(*unknown: Optional[str] = None*)
Validate a constructed model.

exception aries_cloudagent.messaging.models.base.**BaseModelError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Base exception class for base model errors.

class aries_cloudagent.messaging.models.base.**BaseModelSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

BaseModel schema.

class Meta
Bases: object

BaseModelSchema metadata.

model_class = None

ordered = True

skip_values = [None]

property Model: type
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

class aries_cloudagent.messaging.models.base.**SerDe**(*ser*, *de*)
Bases: tuple

property **de**
Alias for field number 1

property **ser**
Alias for field number 0

aries_cloudagent.messaging.models.base.**resolve_class**(*the_cls*, *relative_cls: Optional[type] = None*) → type
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: Optional[str] = None*, *state: Optional[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: Any*, ***kwargs: Any*)
Bases: marshmallow.

Base schema for exchange records.

class **Meta**
Bases: object

BaseExchangeSchema metadata.

model_class
alias of *aries_cloudagent.messaging.models.base_record.BaseExchangeRecord*

trace

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

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

Represents a single storage record.

DEFAULT_CACHE_TTL = 60

EVENT_NAMESPACE: str = 'acapy::record'

LOG_STATE_FLAG = None

class Meta

Bases: object

BaseRecord metadata.

RECORD_ID_NAME = 'id'

RECORD_TOPIC: Optional[str] = None

RECORD_TYPE = None

STATE_DELETED = 'deleted'

TAG_NAMES = {'state'}

async classmethod clear_cached_key(*session:* aries_cloudagent.core.profile.ProfileSession, *cache_key: str*)

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

Parameters

- **session** – The profile session to use
- **cache_key** – The unique cache identifier

async delete_record(*session:* aries_cloudagent.core.profile.ProfileSession)

Remove the stored record.

Parameters **session** – The profile session to use

async emit_event(*session:* aries_cloudagent.core.profile.ProfileSession, *payload: Optional[Any] = None*)

Emit an event.

Parameters

- **session** – The profile session to use
- **payload** – The event payload

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_attributes_by_prefix(*prefix: str*, *walk_mro: bool = True*)
List all values for attributes with common prefix.

Parameters

- **prefix** – Common prefix to look for
- **walk_mro** – Walk MRO to find attributes inherited from superclasses

async classmethod get_cached_key(*session:* aries_cloudagent.core.profile.ProfileSession, *cache_key: str*)
Shortcut method to fetch a cached key value.

Parameters

- **session** – The profile session 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(*msg: str*, *params: Optional[dict] = None*, *settings: Optional[*aries_cloudagent.config.base.BaseSettings*] = None*, *override: bool = False*)
Print a message with increased visibility (for testing).

async post_save(*session:* aries_cloudagent.core.profile.ProfileSession, *new_record: bool*, *last_state: Optional[str]*, *event: Optional[bool] = None*)
Perform post-save actions.

Parameters

- **session** – The profile session to use
- **new_record** – Flag indicating if the record was just created
- **last_state** – The previous state value
- **event** – Flag to override whether the event is sent

classmethod prefix_tag_filter(*tag_filter: dict*)
Prefix unencrypted tags used in the tag filter.

async classmethod query(*session:* aries_cloudagent.core.profile.ProfileSession, *tag_filter: Optional[dict] = None*, ***, *post_filter_positive: Optional[dict] = None*, *post_filter_negative: Optional[dict] = None*, *alt: bool = False*) → Sequence[aries_cloudagent.messaging.models.base_record.RecordType]
Query stored records.

Parameters

- **session** – The profile session 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
- **alt** – set to match any (positive=True) value or miss all (positive=False) values in post_filter

property record_tags: dict
Accessor to define implementation-specific tags.

property record_value: dict
Accessor to define custom properties for the JSON record value.

async classmethod retrieve_by_id(*session:* aries_cloudagent.core.profile.ProfileSession, *record_id: str*, *, *for_update=False*) → aries_cloudagent.messaging.models.base_record.RecordType
Retrieve a stored record by ID.

Parameters

- **session** – The profile session to use
- **record_id** – The ID of the record to find

async classmethod retrieve_by_tag_filter(*session:* aries_cloudagent.core.profile.ProfileSession, *tag_filter: dict*, *post_filter: Optional[dict] = None*, *, *for_update=False*) → aries_cloudagent.messaging.models.base_record.RecordType
Retrieve a record by tag filter.

Parameters

- **session** – The profile session to use
- **tag_filter** – The filter dictionary to apply
- **post_filter** – Additional value filters to apply matching positively, with sequence values specifying alternatives to match (hit any)

async save(*session:* aries_cloudagent.core.profile.ProfileSession, *, *reason: Optional[str] = None*, *log_params: Optional[Mapping[str, Any]] = None*, *log_override: bool = False*, *event: Optional[bool] = None*) → str
Persist the record to storage.

Parameters

- **session** – The profile session to use
- **reason** – A reason to add to the log
- **log_params** – Additional parameters to log
- **override** – Override configured logging regimen, print to stderr instead
- **event** – Flag to override whether the event is sent

async classmethod set_cached_key(*session:* aries_cloudagent.core.profile.ProfileSession, *cache_key: str*, *value: Any*, *ttl=None*)
Shortcut method to set a cached key value.

Parameters

- **session** – The profile session to use
- **cache_key** – The unique cache identifier
- **value** – The value to cache
- **ttl** – The cache ttl

property storage_record: *aries_cloudagent.storage.record.StorageRecord*
Accessor for a *StorageRecord* representing this record.

classmethod strip_tag_prefix(*tags: dict*)
Strip tilde from unencrypted tag names.

property tags: dict
Accessor for the record tags generated for this record.

property value: dict
Accessor for the JSON record value generated for this record.

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

Bases: marshmallow.

Schema to allow serialization/deserialization of base records.

class Meta
Bases: object

BaseRecordSchema metadata.

model_class = None

created_at

state

updated_at

aries_cloudagent.messaging.models.base_record.match_post_filter(record: dict, post_filter: dict, positive: bool = True, alt: bool = False) → 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
- **alt** – set to match any (positive=True) value or miss all (positive=False) values in post_filter

aries_cloudagent.messaging.models.openapi module

Base class for OpenAPI artifact schema.

class aries_cloudagent.messaging.models.openapi.OpenAPISchema(*args: Any, **kwargs: Any)
Bases: marshmallow.

Schema for OpenAPI artifacts: excluding unknown fields, not raising exception.

class Meta
Bases: object

OpenAPISchema metadata.

model_class = None

aries_cloudagent.messaging.schemas package

Submodules

aries_cloudagent.messaging.schemas.routes module

aries_cloudagent.messaging.schemas.util module

Schema utilities.

class aries_cloudagent.messaging.schemas.util.**SchemaQueryStringSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query string parameters for schema searches.

schema_id

schema_issuer_did

schema_name

schema_version

async aries_cloudagent.messaging.schemas.util.**notify_schema_event**(*profile:* aries_cloudagent.core.profile.Profile, *schema_id: str*, *meta_data: dict*)

Send notification for a schema post-process event.

Submodules

aries_cloudagent.messaging.agent_message module

Agent message base class and schema.

class aries_cloudagent.messaging.agent_message.**AgentMessage**(*_id: Optional[str] = None*, *_type: Optional[str] = None*, *_version: Optional[str] = None*, *_decorators: Optional[*aries_cloudagent.messaging.decorators.base.Base = None*)

Bases: *aries_cloudagent.messaging.models.base.BaseModel*, *aries_cloudagent.messaging.base_message.BaseMessage*

Agent message base class.

property Handler: type

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: Optional[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

classmethod deserialize(*value: dict*, *msg_format:* aries_cloudagent.messaging.base_message.DIDCommVersion = *DIDCommVersion.v1*, ***kwargs*)
Return message object deserialized from value in format specified.

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

get_updated_msg_type(*version: str*) → str
Update version to Meta.message_type.

serialize(*msg_format:* aries_cloudagent.messaging.base_message.DIDCommVersion = *DIDCommVersion.v1*, ***kwargs*)
Return serialized message in format specified.

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

async 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

async 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

async verify_signed_field(*field_name: str*, *wallet:* aries_cloudagent.wallet.base.BaseWallet, *signer_verkey: Optional[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: Optional[str] = None*, ***kwargs*)

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

Base exception for agent message issues.

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

Bases: marshmallow.

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*, ***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.

abstract async 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: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Exception base class for generic handler errors.

aries_cloudagent.messaging.base_message module

Base message.

class aries_cloudagent.messaging.base_message.**BaseMessage**
Bases: abc.ABC

Abstract base class for messages.

This formally defines a "minimum viable message" and provides an unopinionated class for plugins to extend in whatever way makes sense in the context of the plugin.

abstract property Handler: Type[BaseHandler]
Return reference to handler class.

abstract classmethod deserialize(*value: dict*, *msg_format:* aries_cloudagent.messaging.base_message.DIDCommVersion = *DIDCommVersion.v1*)
Return message object deserialized from value in format specified.

abstract serialize(*msg_format:* aries_cloudagent.messaging.base_message.DIDCommVersion = *DIDCommVersion.v1*) → dict
Return serialized message in format specified.

class aries_cloudagent.messaging.base_message.**DIDCommVersion**(*value*)
Bases: enum.Enum

Serialized message formats.

v1 = 1

v2 = 2

aries_cloudagent.messaging.error module

Messaging-related error classes and codes.

exception aries_cloudagent.messaging.error.**MessageParseError**(**args*, *error_code: Optional[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: Optional[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**(*profile:* aries_cloudagent.core.profile.Profile, **, context: Optional[*aries_cloudagent.config.injection_context.Inj *= None*, *settings: Optional[Mapping[str, object]] = None*)

Bases: object

Context established by the Conductor and passed into message handlers.

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

Returns True if the connection is active, else False

property connection_record: Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]
Accessor for the related connection record.

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

Returns The default agent endpoint

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

Returns The default label

inject(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*) → aries_cloudagent.config.base.InjectType
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

inject_or(*base_cls: Type[aries_cloudagent.config.base.InjectType]*, *settings: Optional[Mapping[str, object]] = None*, *default: Optional[aries_cloudagent.config.base.InjectType] = None*) → Optional[aries_cloudagent.config.base.InjectType]

Get the provided instance of a given class identifier or default if not found.

Parameters

- **base_cls** – The base class to retrieve an instance of
- **settings** – An optional dict providing configuration to the provider
- **default** – default return value if no instance is found

Returns An instance of the base class, or None

property injector: *aries_cloudagent.config.injector.Injector*

Accessor for the associated *Injector* instance.

property message: *aries_cloudagent.messaging.agent_message.AgentMessage*

Accessor for the deserialized message instance.

Returns This context's agent message

property message_receipt: *aries_cloudagent.transport.inbound.receipt.MessageReceipt*

Accessor for the message receipt information.

Returns This context's message receipt information

property profile: *aries_cloudagent.core.profile.Profile*

Accessor for the associated *Profile* instance.

session() → *aries_cloudagent.core.profile.ProfileSession*

Start a new interactive session with no transaction support requested.

property settings: *aries_cloudagent.config.settings.Settings*

Accessor for the context settings.

classmethod test_context() → *aries_cloudagent.messaging.request_context.RequestContext*

Quickly set up a new request context for tests.

transaction() → *aries_cloudagent.core.profile.ProfileSession*

Start a new interactive session with commit and rollback support.

If the current backend does not support transactions, then commit and rollback operations of the session will not have any effect.

update_settings(*settings: Mapping[str, object]*)

Update the scope with additional settings.

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: Optional[str] = None*, *reply_session_id: Optional[str] = None*, *reply_to_verkey: Optional[str] = None*)

Bases: abc.ABC

Interface for message handlers to send responses.

async conn_rec_active_state_check(*profile:* aries_cloudagent.core.profile.Profile, *connection_id: str*, *timeout: int = 7*) → bool

Check if the connection record is ready for sending outbound message.

async create_outbound(*message: Union[*aries_cloudagent.messaging.base_message.BaseMessage*, str, bytes]*, *, *connection_id: Optional[str] = None*, *reply_session_id: Optional[str] = None*, *reply_thread_id: Optional[str] = None*, *reply_to_verkey: Optional[str] = None*, *reply_from_verkey: Optional[str] = None*, *target: Optional[*aries_cloudagent.connections.models.connection_target.ConnectionTarget*] = None*, *target_list: Optional[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.

async send(*message: Union[*aries_cloudagent.messaging.base_message.BaseMessage*, str, bytes]*, ***kwargs*) → *aries_cloudagent.transport.outbound.status.OutboundSendStatus*

Convert a message to an OutboundMessage and send it.

abstract async send_outbound(*message:* aries_cloudagent.transport.outbound.message.OutboundMessage, ***kwargs*) → *aries_cloudagent.transport.outbound.status.OutboundSendStatus*

Send an outbound message.

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

async send_reply(*message: Union[*aries_cloudagent.messaging.base_message.BaseMessage*, str, bytes]*, *, *connection_id: Optional[str] = None*, *target: Optional[*aries_cloudagent.connections.models.connection_target.ConnectionTarget*] = None*, *target_list: Optional[Sequence[*aries_cloudagent.connections.models.connection_target.ConnectionTarget*]] = None*) → *aries_cloudagent.transport.outbound.status.OutboundSendStatus*

Send a reply to an incoming message.

Parameters

- **message** – the *BaseMessage*, 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

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

Dispatch a webhook. DEPRECATED: use the event bus instead.

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.

async **send**(*message: Union[*aries_cloudagent.messaging.base_message.BaseMessage*, str, bytes]*, ***kwargs*) → *aries_cloudagent.transport.outbound.status.OutboundSendStatus*

Convert a message to an OutboundMessage and send it.

async **send_outbound**(*message:* aries_cloudagent.transport.outbound.message.OutboundMessage, ***kwargs*) → *aries_cloudagent.transport.outbound.status.OutboundSendStatus*

Send an outbound message.

async **send_reply**(*message: Union[*aries_cloudagent.messaging.base_message.BaseMessage*, str, bytes]*, ***kwargs*) → *aries_cloudagent.transport.outbound.status.OutboundSendStatus*

Send a reply to an incoming message.

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

Send an outbound message.

exception aries_cloudagent.messaging.responder.**ResponderError**(**args*, *error_code: Optional[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**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against base58 encoding of SHA-256 hash.

EXAMPLE = 'H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV'

PATTERN = '^[base58.alphabet.decode]{43,44}$'

class aries_cloudagent.messaging.valid.**Base64**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate base64 value.

EXAMPLE = 'ey4uLn0='

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

class aries_cloudagent.messaging.valid.**Base64URL**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate base64 value.

EXAMPLE = 'ey4uLn0='

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

class aries_cloudagent.messaging.valid.**Base64URLNoPad**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate base64 value.

EXAMPLE = 'ey4uLn0'

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

class aries_cloudagent.messaging.valid.**CredentialContext**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Credential Context.

EXAMPLE = ['https://www.w3.org/2018/credentials/v1', 'https://www.w3.org/2018/credentials/examples/v1']

FIRST_CONTEXT = 'https://www.w3.org/2018/credentials/v1'

class aries_cloudagent.messaging.valid.**CredentialSubject**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Credential subject.

EXAMPLE = {'alumniOf': {'id': 'did:example:c276e12ec21ebfeb1f712ebc6f1'}, 'id': 'did:example:ebfeb1f712ebc6f1c276e12ec21'}

class aries_cloudagent.messaging.valid.**CredentialType**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Credential Type.

CREDENTIAL_TYPE = 'VerifiableCredential'

EXAMPLE = ['VerifiableCredential', 'AlumniCredential']

class aries_cloudagent.messaging.valid.**DIDKey**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against DID key specification.

EXAMPLE = 'did:key:z6MkpTHR8VNsBxYAAWHut2Geadd9jSwuBV8xRoAnwWsdvktH'

PATTERN = re.compile('^did:key:z[base58.alphabet.decode]+$')

class aries_cloudagent.messaging.valid.**DIDPosture**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against defined DID postures.

EXAMPLE = 'wallet_only'

class aries_cloudagent.messaging.valid.**DIDValidation**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against any valid DID spec.

EXAMPLE = 'did:peer:WgWxqztrNooG92RXvxSTWv'

FRAGMENT = '(\\#.*)?$'

METHOD = '([a-zA-Z0-9_]+)'

METHOD_ID = '([a-zA-Z0-9_.%-]+(:[a-zA-Z0-9_.%-]+)*)'

PARAMS = '((;[a-zA-Z0-9_.:%-]+=[a-zA-Z0-9_.:%-]*)*)'

PATH = '(\\/[^#?]*)?'

PATTERN = re.compile('^did:([a-zA-Z0-9_]+):([a-zA-Z0-9_.%-]+(:[a-zA-Z0-9_.%-]+)*)((;[a-zA-Z0-9_.:%-]+=[a-zA-Z0-9_.:%-]*)*)(\\/[^#?]*)?([?][^#]*)?(\\#.*)?$$')

QUERY = '([?][^#]*)?'

class aries_cloudagent.messaging.valid.**DIDWeb**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against did:web specification.

EXAMPLE = 'did:web:example.com'

PATTERN = re.compile('^(did:web:)([a-zA-Z0-9%._-]*:)*[a-zA-Z0-9%._-]+$')

class aries_cloudagent.messaging.valid.**DictOrDictListField**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.fields.

Dict or Dict List field for Marshmallow.

class aries_cloudagent.messaging.valid.**Endpoint**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against endpoint URL on any scheme.

EXAMPLE = 'https://myhost:8021'

PATTERN = '^[A-Za-z0-9\\.\\-\\+]+://([A-Za-z0-9][.
A-Za-z0-9-_]+[A-Za-z0-9])+(:[1-9][0-9]*)?(/[^?&#]+)?$'

class aries_cloudagent.messaging.valid.**EndpointType**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against allowed endpoint/service types.

EXAMPLE = 'Endpoint'

class aries_cloudagent.messaging.valid.**IndyCredDefId**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy credential definition identifier specification.

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

PATTERN = '^([base58.alphabet.decode]{21,22}):3:CL:(([1-9][0-9]*)|([base58.alphabet.
decode]{21,22}:2:.+:[0-9.]+)):(.+)?$'

class aries_cloudagent.messaging.valid.**IndyCredRevId**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy credential revocation identifier specification.

EXAMPLE = '12345'

PATTERN = '^[1-9][0-9]*$'

class aries_cloudagent.messaging.valid.**IndyDID**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy DID.

EXAMPLE = 'WgWxqztrNooG92RXvxSTWv'

PATTERN = re.compile('^(did:sov:)?[base58.alphabet.decode]{21,22}$')

class aries_cloudagent.messaging.valid.**IndyExtraWQL**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value as potential extra WQL query in cred search for proof req.

EXAMPLE = '{"0_drink_uuid": {"attr::drink::value": "martini"}}'

PATTERN = '^{\\s*".*?"\\s*:\\s*{.*?}\\s*(,\\s*".*?"\\s*:\\s*{.*?}\\s*)*\\s*}$'

class aries_cloudagent.messaging.valid.**IndyISO8601DateTime**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against ISO 8601 datetime format, indy profile.

EXAMPLE = '2021-12-31T23:59:59Z'

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

class aries_cloudagent.messaging.valid.**IndyOrKeyDID**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Indy or Key DID class.

EXAMPLE = 'WgWxqztrNooG92RXvxSTWv'

PATTERN = '^did:key:z[base58.alphabet.decode]+$|^(did:sov:)?[base58.alphabet.decode]{21,22}$'

class aries_cloudagent.messaging.valid.**IndyPredicate**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy predicate.

EXAMPLE = '>='

class aries_cloudagent.messaging.valid.**IndyRawPublicKey**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy (Ed25519VerificationKey2018) raw public key.

EXAMPLE = 'H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV'

PATTERN = '^[base58.alphabet.decode]{43,44}$'

class aries_cloudagent.messaging.valid.**IndyRevRegId**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy revocation registry identifier specification.

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

PATTERN = '^([base58.alphabet.decode]{21,22}):4:([base58.alphabet.decode]{21,22}):3:CL:(([1-9][0-9]*)|([base58.alphabet.decode]{21,22}:2:.+:[0-9.]+))(:.+)?:CL_ACCUM:(.+$)'

class aries_cloudagent.messaging.valid.**IndyRevRegSize**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value as indy revocation registry size.

EXAMPLE = 1000

class aries_cloudagent.messaging.valid.**IndySchemaId**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy schema identifier specification.

EXAMPLE = 'WgWxqztrNooG92RXvxSTWv:2:schema_name:1.0'

PATTERN = '^[base58.alphabet.decode]{21,22}:2:.+:[0-9.]+$'

class aries_cloudagent.messaging.valid.**IndyVersion**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against indy version specification.

EXAMPLE = '1.0'

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

class aries_cloudagent.messaging.valid.IndyWQL(*args: Any, **kwargs: Any)
Bases: marshmallow.validate.

Validate value as potential WQL query.

EXAMPLE = '{"attr::name::value": "Alex"}'

PATTERN = '^{.*}$'

class aries_cloudagent.messaging.valid.IntEpoch(*args: Any, **kwargs: Any)
Bases: marshmallow.validate.

Validate value against (integer) epoch format.

EXAMPLE = 1640995199

class aries_cloudagent.messaging.valid.JSONWebToken(*args: Any, **kwargs: Any)
Bases: marshmallow.validate.

Validate JSON Web Token.

EXAMPLE =
'eyJhbGciOiJFZERTQSJ9.eyJhIjogIjAifQ.dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk'

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

class aries_cloudagent.messaging.valid.JWSHeaderKid(*args: Any, **kwargs: Any)
Bases: marshmallow.validate.

Validate value against JWS header kid.

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

PATTERN = '^did:(?:key:z[base58.alphabet.decode]+|sov:[base58.alphabet.decode]{21,
22}(;.*)?(\\?.*)?#.+)$'

class aries_cloudagent.messaging.valid.MaybeIndyDID(*args: Any, **kwargs: Any)
Bases: marshmallow.validate.

Validate value against any valid DID spec or a short Indy DID.

EXAMPLE = 'did:peer:WgWxqztrNooG92RXvxSTWv'

PATTERN = re.compile('^(did:sov:)?[base58.alphabet.decode]{21,
22}$|^did:([a-zA-Z0-9_]+):([a-zA-Z0-9_.%-]+(:[a-zA-Z0-9_.%-]+)*)((;[a-zA-Z0-9_.
:%-]+=[a-zA-Z0-9_.:%-]*)*)(\\/[^#?]*)?([?][^#]*)?(\\#.*)?$$')

class aries_cloudagent.messaging.valid.NaturalNumber(*args: Any, **kwargs: Any)
Bases: marshmallow.validate.

Validate value as positive integer.

EXAMPLE = 10

class aries_cloudagent.messaging.valid.NumericStrAny(*args: Any, **kwargs: Any)
Bases: marshmallow.validate.

Validate value against any number numeric string.

EXAMPLE = '-1'

PATTERN = '^-?[0-9]*$'

class aries_cloudagent.messaging.valid.**NumericStrNatural**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against natural number numeric string.

EXAMPLE = '1'

PATTERN = '^[1-9][0-9]*$'

class aries_cloudagent.messaging.valid.**NumericStrWhole**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against whole number numeric string.

EXAMPLE = '0'

PATTERN = '^[0-9]*$'

class aries_cloudagent.messaging.valid.**RFC3339DateTime**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against RFC3339 datetime format.

EXAMPLE = '2010-01-01T19:23:24Z'

PATTERN = '^([0-9]{4})-([0-9]{2})-([0-9]{2})([Tt]([0-9]{2}):([0-9]{2}):([0-9]{2})(
\.[0-9]+)?)?(([Zz]|([+-])([0-9]{2}):([0-9]{2})))?$'

class aries_cloudagent.messaging.valid.**RoutingKey**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate between indy or did key.

Validate value against indy (Ed25519VerificationKey2018) raw public key or DID key specification.

EXAMPLE = 'did:key:z6MkpTHR8VNsBxYAAWHut2Geadd9jSwuBV8xRoAnwWsdvktH'

PATTERN =
re.compile('^did:key:z[base58.alphabet.decode]+$|^[base58.alphabet.decode]{43,44}$')

class aries_cloudagent.messaging.valid.**SHA256Hash**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate (binhex-encoded) SHA256 value.

EXAMPLE = '617a48c7c8afe0521efdc03e5bb0ad9e655893e6b4b51f0e794d70fba132aacb'

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

class aries_cloudagent.messaging.valid.**StrOrDictField**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.fields.

URI or Dict field for Marshmallow.

class aries_cloudagent.messaging.valid.**StrOrNumberField**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.fields.

String or Number field for Marshmallow.

class aries_cloudagent.messaging.valid.**UUIDFour**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

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}'

class aries_cloudagent.messaging.valid.**Uri**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against URI on any scheme.

EXAMPLE = 'https://www.w3.org/2018/credentials/v1'

PATTERN = '\\w+:(\\/?\\/?)[^\\s]+'

class aries_cloudagent.messaging.valid.**UriOrDictField**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.fields.

URI or Dict field for Marshmallow.

class aries_cloudagent.messaging.valid.**WholeNumber**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value as non-negative integer.

EXAMPLE = 0

aries_cloudagent.multitenant package

Subpackages

aries_cloudagent.multitenant.admin package

Submodules

aries_cloudagent.multitenant.admin.routes module

Submodules

aries_cloudagent.multitenant.askar_profile_manager module

aries_cloudagent.multitenant.base module

aries_cloudagent.multitenant.cache module

Cache for multitenancy profiles.

class aries_cloudagent.multitenant.cache.**ProfileCache**(*capacity: int*)
Bases: object

Profile cache that caches based on LRU strategy.

get(*key: str*) → Optional[*aries_cloudagent.core.profile.Profile*]
Get profile with associated key from cache.

If a profile is open but has been evicted from the cache, this will reinsert the profile back into the cache. This prevents attempting to open a profile that is already open. Triggers clean up.

Parameters **key** (*str*) – the key to get the profile for.

Returns Profile if found in cache.

Return type Optional[Profile]

has(*key: str*) → bool
Check whether there is a profile with associated key in the cache.

Parameters **key** (*str*) – the key to check for a profile

Returns Whether the key exists in the cache

Return type bool

put(*key: str*, *value:* aries_cloudagent.core.profile.Profile) → None
Add profile with associated key to the cache.

If new profile exceeds the cache capacity least recently used profiles that are not used will be removed from the cache.

Parameters

- **key** (*str*) – the key to set
- **value** (*Profile*) – the profile to set

remove(*key: str*)
Remove profile with associated key from the cache.

Parameters **key** (*str*) – The key to remove from the cache.

aries_cloudagent.multitenant.error module

Multitenant error classes.

exception aries_cloudagent.multitenant.error.**WalletKeyMissingError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Wallet key missing exception.

aries_cloudagent.multitenant.manager module

aries_cloudagent.multitenant.manager_provider module

Profile manager for multitenancy.

class aries_cloudagent.multitenant.manager_provider.**MultitenantManagerProvider**(*root_profile*)

Bases: *aries_cloudagent.config.base.BaseProvider*

Multitenant manager provider.

Decides which manager to use based on the settings.

MANAGER_TYPES = {'askar-profile':
'aries_cloudagent.multitenant.askar_profile_manager.AskarProfileMultitenantManager',
'basic': 'aries_cloudagent.multitenant.manager.MultitenantManager'}

askar_profile_manager_path =
'aries_cloudagent.multitenant.askar_profile_manager.AskarProfileMultitenantManager'

provide(*settings:* aries_cloudagent.config.base.BaseSettings, *injector:* aries_cloudagent.config.base.BaseInjector)
Create the multitenant manager instance.

aries_cloudagent.multitenant.route_manager module

aries_cloudagent.protocols package

Subpackages

aries_cloudagent.protocols.actionmenu package

Subpackages

aries_cloudagent.protocols.actionmenu.v1_0 package

Subpackages

aries_cloudagent.protocols.actionmenu.v1_0.handlers package

Submodules

aries_cloudagent.protocols.actionmenu.v1_0.handlers.menu_handler module

Action menu message handler.

class aries_cloudagent.protocols.actionmenu.v1_0.handlers.menu_handler.**MenuHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for action menus.

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

Message handler logic for action menus.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.actionmenu.v1_0.handlers.menu_request_handler module

Action menu request message handler.

class aries_cloudagent.protocols.actionmenu.v1_0.handlers.menu_request_handler.**MenuRequestHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for action menu requests.

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

Message handler logic for action menu requests.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.actionmenu.v1_0.handlers.perform_handler module

Action menu perform request message handler.

class aries_cloudagent.protocols.actionmenu.v1_0.handlers.perform_handler.**PerformHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for action menu perform requests.

async handle(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Message handler logic for action menu perform requests.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.actionmenu.v1_0.messages package

Submodules

aries_cloudagent.protocols.actionmenu.v1_0.messages.menu module

Represents an action menu.

class aries_cloudagent.protocols.actionmenu.v1_0.messages.menu.**Menu**(*, *title: Optional[str] = None*, *description: Optional[str] = None*, *errormsg: Optional[str] = None*, *options: Optional[Sequence[*aries_cloudagent.protocols.ac *= None*, ***kwargs*)
Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing an action menu.

class Meta
Bases: object

Metadata for an action menu.

handler_class = 'aries_cloudagent.protocols.actionmenu.v1_0.handlers.menu_handler.MenuHandler'

message_type = 'action-menu/1.0/menu'

schema_class = 'MenuSchema'

class aries_cloudagent.protocols.actionmenu.v1_0.messages.menu.**MenuSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Menu schema class.

class Meta
Bases: object

Menu schema metadata.

model_class
alias of *aries_cloudagent.protocols.actionmenu.v1_0.messages.menu.Menu*

aries_cloudagent.protocols.actionmenu.v1_0.messages.menu_request module

Represents a request for an action menu.

class aries_cloudagent.protocols.actionmenu.v1_0.messages.menu_request.**MenuRequest**(***kwargs*)
Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a request for an action menu.

class Meta
Bases: object

Metadata for action menu request.

handler_class = 'aries_cloudagent.protocols.actionmenu.v1_0.handlers.menu_request_handler.MenuRequestHandler'

message_type = 'action-menu/1.0/menu-request'

schema_class = 'MenuRequestSchema'

class aries_cloudagent.protocols.actionmenu.v1_0.messages.menu_request.**MenuRequestSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

MenuRequest schema class.

class Meta
Bases: object

MenuRequest schema metadata.

model_class
alias of *aries_cloudagent.protocols.actionmenu.v1_0.messages.menu_request.MenuRequest*

aries_cloudagent.protocols.actionmenu.v1_0.messages.perform module

Represents a request to perform a menu action.

class aries_cloudagent.protocols.actionmenu.v1_0.messages.perform.**Perform**(*, *name: Optional[str] = None*, *params: Optional[Mapping[str, str]] = None*, ***kwargs*)
Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a request to perform a menu action.

class Meta
Bases: object

Perform metadata.

handler_class = 'aries_cloudagent.protocols.actionmenu.v1_0.handlers.perform_handler.PerformHandler'

message_type = 'action-menu/1.0/perform'

schema_class = 'PerformSchema'

class aries_cloudagent.protocols.actionmenu.v1_0.messages.perform.**PerformSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Perform schema class.

class Meta

Bases: object

Perform schema metadata.

model_class

alias of *aries_cloudagent.protocols.actionmenu.v1_0.messages.perform.Perform*

name

params

aries_cloudagent.protocols.actionmenu.v1_0.models package

Submodules

aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form module

Record used to represent the form associated with an action menu option.

class aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form.**MenuForm**(***, *title: Optional[str] = None*, *description: Optional[str] = None*, *params: Optional[Sequence[*aries_cloudagent.pr *= None*, *submit_label: Optional[str] = None*)

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

Instance of a form associated with an action menu item.

class Meta

Bases: object

Menu form metadata.

schema_class = 'MenuFormSchema'

class aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form.**MenuFormSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

MenuForm schema.

class **Meta**

Bases: object

MenuFormSchema metadata.

model_class

alias of *aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form.MenuForm*

aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form_param module

Record used to represent a parameter in a menu form.

class aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form_param.**MenuFormParam**(*, *name: Optional[str] = None*, *title: Optional[str] = None*, *default: Optional[str] = None*, *description: Optional[str] = None*, *input_type: Optional[str] = None*, *required: Optional[bool] = None*)

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

Instance of a menu form param associated with an action menu option.

class Meta

Bases: object

Menu form param metadata.

schema_class = 'MenuFormParamSchema'

class aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form_param.**MenuFormParamSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

MenuFormParam schema.

class Meta

Bases: object

MenuFormParamSchema metadata.

model_class
alias of *aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form_param.MenuFormParam*

aries_cloudagent.protocols.actionmenu.v1_0.models.menu_option module

Record used to represent individual menu options in an action menu.

class aries_cloudagent.protocols.actionmenu.v1_0.models.menu_option.**MenuOption**(*, *name: Optional[str] = None*, *title: Optional[str] = None*, *description: Optional[str] = None*, *disabled: Optional[bool] = None*, *form: Optional[*aries_cloudagent.protocol *= None*)

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

Instance of a menu option associated with an action menu.

class Meta
Bases: object

Menu option metadata.

schema_class = 'MenuOptionSchema'

class aries_cloudagent.protocols.actionmenu.v1_0.models.menu_option.**MenuOptionSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

MenuOption schema.

class Meta
Bases: object

MenuOptionSchema metadata.

model_class
alias of *aries_cloudagent.protocols.actionmenu.v1_0.models.menu_option.MenuOption*

form
MenuForm schema.

Submodules

aries_cloudagent.protocols.actionmenu.v1_0.base_service module

Base action menu service classes.

class aries_cloudagent.protocols.actionmenu.v1_0.base_service.**BaseMenuService**(*context:* aries_cloudagent.config.injection

Bases: abc.ABC

Base action menu service interface.

abstract async get_active_menu(*profile:* aries_cloudagent.core.profile.Profile, *connection: Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]* = *None*, *thread_id: Optional[str]* = *None*) → *aries_cloudagent.protocols.actionmenu.v1_0.messages.menu.Menu*

Render the current menu.

Parameters

- **profile** – The profile
- **connection** – The active connection record
- **thread_id** – The thread identifier from the requesting message.

abstract async perform_menu_action(*profile:* aries_cloudagent.core.profile.Profile, *action_name: str*, *action_params: dict*, *connection: Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]* = *None*, *thread_id: Optional[str]* = *None*) → *aries_cloudagent.messaging.agent_message.AgentMessage*

Perform an action defined by the active menu.

Parameters

- **profile** – The profile
- **action_name** – The unique name of the action being performed
- **action_params** – A collection of parameters for the action
- **connection** – The active connection record
- **thread_id** – The thread identifier from the requesting message.

classmethod service_handler()

Quick accessor for conductor to use.

aries_cloudagent.protocols.actionmenu.v1_0.controller module

Protocol controller for the action menu message family.

class aries_cloudagent.protocols.actionmenu.v1_0.controller.**Controller**(*protocol: str*)

Bases: object

Action menu protocol controller.

determine_goal_codes() → Sequence[str]

Return defined goal_codes.

async determine_roles(*context:* aries_cloudagent.config.injection_context.InjectionContext) → Sequence[str]

Determine what action menu roles are defined.

aries_cloudagent.protocols.actionmenu.v1_0.driver_service module

Driver-based action menu service classes.

class aries_cloudagent.protocols.actionmenu.v1_0.driver_service.DriverMenuService(*context:* aries_cloudagent.config.inje

Bases: *aries_cloudagent.protocols.actionmenu.v1_0.base_service.BaseMenuService*

Driver-based action menu service.

async get_active_menu(*profile:* aries_cloudagent.core.profile.Profile, *connection:* *Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]* = *None*, *thread_id: Optional[str]* = *None*) → *aries_cloudagent.protocols.actionmenu.v1_0.messages.menu.Menu*

Render the current menu.

Parameters

- **profile** – The profile
- **connection** – The active connection record
- **thread_id** – The thread identifier from the requesting message.

async perform_menu_action(*profile:* aries_cloudagent.core.profile.Profile, *action_name: str*, *action_params: dict*, *connection:* *Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]* = *None*, *thread_id: Optional[str]* = *None*) → *aries_cloudagent.messaging.agent_message.AgentMessage*

Perform an action defined by the active menu.

Parameters

- **profile** – The profile
- **action_name** – The unique name of the action being performed
- **action_params** – A collection of parameters for the action
- **connection** – The active connection record
- **thread_id** – The thread identifier from the requesting message.

aries_cloudagent.protocols.actionmenu.v1_0.message_types module

Message type identifiers for Action Menus.

aries_cloudagent.protocols.actionmenu.v1_0.routes module

Action menu admin routes.

class aries_cloudagent.protocols.actionmenu.v1_0.routes.**ActionMenuFetchResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for action-menu fetch.

result

class aries_cloudagent.protocols.actionmenu.v1_0.routes.**ActionMenuModulesResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for the modules endpoint.

class aries_cloudagent.protocols.actionmenu.v1_0.routes.**MenuConnIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters and validators for request taking connection id.

conn_id

class aries_cloudagent.protocols.actionmenu.v1_0.routes.**MenuJsonSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Matches MenuSchema but without the inherited AgentMessage properties.

description

errormsg

options

title

class aries_cloudagent.protocols.actionmenu.v1_0.routes.**PerformRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for performing a menu action.

name

params

class aries_cloudagent.protocols.actionmenu.v1_0.routes.**SendMenuSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending a menu to a connection.

menu

aries_cloudagent.protocols.actionmenu.v1_0.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.actionmenu.v1_0.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

aries_cloudagent.protocols.actionmenu.v1_0.util module

Action menu utility methods.

async aries_cloudagent.protocols.actionmenu.v1_0.util.**retrieve_connection_menu**(*connection_id: str*, *context: aries_cloudagent.admin.request_* → *aries_cloudagent.protocols.acti*

Retrieve the previously-received action menu.

async aries_cloudagent.protocols.actionmenu.v1_0.util.**save_connection_menu**(*menu: aries_cloudagent.protocols.actionmen connection_id: str*, *context: aries_cloudagent.admin.request_cont*

Save a received action menu.

Submodules

aries_cloudagent.protocols.actionmenu.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.basicmessage package

Subpackages

aries_cloudagent.protocols.basicmessage.v1_0 package

Subpackages

aries_cloudagent.protocols.basicmessage.v1_0.handlers package

Submodules

aries_cloudagent.protocols.basicmessage.v1_0.handlers.basicmessage_handler module

Basic message handler.

class aries_cloudagent.protocols.basicmessage.v1_0.handlers.basicmessage_handler.**BasicMessageHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for basic messages.

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

Message handler logic for basic messages.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.basicmessage.v1_0.messages package

Submodules

aries_cloudagent.protocols.basicmessage.v1_0.messages.basicmessage module

Basic message.

class aries_cloudagent.protocols.basicmessage.v1_0.messages.basicmessage.**BasicMessage**(*, *sent_time: Optional[Union[str, datetime.datetime]] = None*, *content: Optional[str] = None*, *localization: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class defining the structure of a basic message.

class Meta

Bases: object

Basic message metadata class.

handler_class = 'aries_cloudagent.protocols.basicmessage.v1_0.handlers.basicmessage_handler.BasicMessageHandler'

message_type = 'basicmessage/1.0/message'

schema_class = 'BasicMessageSchema'

class aries_cloudagent.protocols.basicmessage.v1_0.messages.basicmessage.**BasicMessageSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Basic message schema class.

class Meta

Bases: object

Basic message schema metadata.

model_class

alias of *aries_cloudagent.protocols.basicmessage.v1_0.messages.basicmessage.BasicMessage*

content

sent_time

Submodules

aries_cloudagent.protocols.basicmessage.v1_0.message_types module

Message type identifiers for Connections.

aries_cloudagent.protocols.basicmessage.v1_0.routes module

Basic message admin routes.

class aries_cloudagent.protocols.basicmessage.v1_0.routes.**BasicConnIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters and validators for request taking connection id.

conn_id

class aries_cloudagent.protocols.basicmessage.v1_0.routes.**BasicMessageModuleResponseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Response schema for Basic Message Module.

class aries_cloudagent.protocols.basicmessage.v1_0.routes.**SendMessageSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending a message.

content

aries_cloudagent.protocols.basicmessage.v1_0.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.basicmessage.v1_0.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

Submodules

aries_cloudagent.protocols.basicmessage.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.connections package

Subpackages

aries_cloudagent.protocols.connections.v1_0 package

Subpackages

aries_cloudagent.protocols.connections.v1_0.handlers package

Submodules

aries_cloudagent.protocols.connections.v1_0.handlers.connection_invitation_handler module

Connect invitation handler.

class aries_cloudagent.protocols.connections.v1_0.handlers.connection_invitation_handler.**ConnectionInvitationHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for connection invitations.

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

Handle connection invitation.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.connections.v1_0.handlers.connection_request_handler module

aries_cloudagent.protocols.connections.v1_0.handlers.connection_response_handler module

aries_cloudagent.protocols.connections.v1_0.messages package

Submodules

aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation module

Represents an invitation message for establishing connection.

class aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.**ConnectionInvitation**(*,

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a connection invitation.

class Meta

Bases: object

Metadata for a connection invitation.

handler_class = 'aries_cloudagent.protocols.connections.v1_0.handlers.connection_invitation_handler.ConnectionInvitationHandler'

message_type = 'connections/1.0/invitation'

schema_class = 'ConnectionInvitationSchema'

classmethod from_url(*url: str*) → *aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.ConnectionInvitation*

Parse a URL-encoded invitation into a *ConnectionInvitation* message.

Parameters **url** – Url to decode

Returns A *ConnectionInvitation* object.

to_url(*base_url: Optional[str] = None*) → str

Convert an invitation to URL format for sharing.

Returns An invite url

class aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.**ConnectionInvitationSch**

Bases: marshmallow.

Connection invitation schema class.

class Meta

Bases: object

Connection invitation schema metadata.

model_class

alias of *aries_cloudagent.protocols.connections.v1_0.messages.connection_invitation.ConnectionInvitation*

did

endpoint

image_url

label

recipient_keys

routing_keys

validate_fields(*data*, ***kwargs*)

Validate schema fields.

Parameters **data** – The data to validate

Raises **ValidationError** – If any of the fields do not validate

aries_cloudagent.protocols.connections.v1_0.messages.connection_request module

Represents a connection request message.

class aries_cloudagent.protocols.connections.v1_0.messages.connection_request.**ConnectionRequest**(*, *connection: Optional[ari* = *None*, *label: Optional[str* = *None*, *image_url: Optional[str* = *None*, ***kwargs*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a connection request.

class Meta

Bases: object

Metadata for a connection request.

handler_class = 'aries_cloudagent.protocols.connections.v1_0.handlers.connection_request_handler.ConnectionRequestHandler'

message_type = 'connections/1.0/request'

schema_class = 'ConnectionRequestSchema'

class aries_cloudagent.protocols.connections.v1_0.messages.connection_request.**ConnectionRequestSchema**(*

A
**
A

Bases: marshmallow.

Connection request schema class.

class Meta

Bases: object

Connection request schema metadata.

model_class

alias of *aries_cloudagent.protocols.connections.v1_0.messages.connection_request.ConnectionRequest*

connection
alias of *aries_cloudagent.protocols.connections.v1_0.models.connection_detail.ConnectionDetailSchema*

aries_cloudagent.protocols.connections.v1_0.messages.connection_response module

Represents a connection response message.

class aries_cloudagent.protocols.connections.v1_0.messages.connection_response.**ConnectionResponse**(*, *connection: Optional[* = *None*, ***kwa*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a connection response.

class Meta
Bases: object

Metadata for a connection response.

handler_class = 'aries_cloudagent.protocols.connections.v1_0.handlers.connection_response_handler.ConnectionResponseHandler'

message_type = 'connections/1.0/response'

schema_class = 'ConnectionResponseSchema'

class aries_cloudagent.protocols.connections.v1_0.messages.connection_response.**ConnectionResponseSchema**

Bases: marshmallow.

Connection response schema class.

class Meta
Bases: object

Connection response schema metadata.

model_class
alias of *aries_cloudagent.protocols.connections.v1_0.messages.connection_response.ConnectionResponse*

signed_fields = ('connection',)

connection

aries_cloudagent.protocols.connections.v1_0.messages.problem_report module

Represents a connection problem report message.

class aries_cloudagent.protocols.connections.v1_0.messages.problem_report.**ConnectionProblemReport**(*, *problem_co* *Optional[* = *None*, *explain: Optional[* = *None*, ***kwa*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Base class representing a connection problem report message.

class Meta

Bases: object

Connection problem report metadata.

handler_class = 'aries_cloudagent.protocols.problem_report.v1_0.handler.ProblemReportHandler'

message_type = 'connections/1.0/problem_report'

schema_class = 'ConnectionProblemReportSchema'

class aries_cloudagent.protocols.connections.v1_0.messages.problem_report.**ConnectionProblemReportSchema**

Bases: marshmallow.

Schema for ConnectionProblemReport base class.

class Meta

Bases: object

Metadata for connection problem report schema.

model_class

alias of *aries_cloudagent.protocols.connections.v1_0.messages.problem_report.ConnectionProblemReport*

explain

problem_code

class aries_cloudagent.protocols.connections.v1_0.messages.problem_report.**ProblemReportReason**(*value*)

Bases: enum.Enum

Supported reason codes.

INVITATION_NOT_ACCEPTED = 'invitation_not_accepted'

REQUEST_NOT_ACCEPTED = 'request_not_accepted'

REQUEST_PROCESSING_ERROR = 'request_processing_error'

RESPONSE_NOT_ACCEPTED = 'response_not_accepted'

RESPONSE_PROCESSING_ERROR = 'response_processing_error'

aries_cloudagent.protocols.connections.v1_0.models package

Submodules

aries_cloudagent.protocols.connections.v1_0.models.connection_detail module

An object for containing the connection request/response DID information.

class aries_cloudagent.protocols.connections.v1_0.models.connection_detail.**ConnectionDetail**(*, *did: Optional[str] = None*, *did_doc: Optional[aries_cl = None*, ***kwargs*)

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

Class representing the details of a connection.

class **Meta**

Bases: object

ConnectionDetail metadata.

schema_class = 'ConnectionDetailSchema'

property **did**: str

Accessor for the connection DID.

Returns The DID for this connection

property **did_doc**: *aries_cloudagent.connections.models.diddoc.diddoc.DIDDoc*

Accessor for the connection DID Document.

Returns The DIDDoc for this connection

class aries_cloudagent.protocols.connections.v1_0.models.connection_detail.**ConnectionDetailSchema**(**args: Any*, ***kwa Any*)

Bases: marshmallow.

ConnectionDetail schema.

class **Meta**

Bases: object

ConnectionDetailSchema metadata.

model_class
alias of *aries_cloudagent.protocols.connections.v1_0.models.connection_detail.ConnectionDetail*

did_doc
Field that loads and serializes DIDDoc.

class aries_cloudagent.protocols.connections.v1_0.models.connection_detail.**DIDDocWrapper**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.fields.

Field that loads and serializes DIDDoc.

Submodules

aries_cloudagent.protocols.connections.v1_0.manager module

aries_cloudagent.protocols.connections.v1_0.message_types module

Message type identifiers for Connections.

aries_cloudagent.protocols.connections.v1_0.routes module

Submodules

aries_cloudagent.protocols.connections.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.coordinate_mediation package

Subpackages

aries_cloudagent.protocols.coordinate_mediation.v1_0 package

Subpackages

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers package

Submodules

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_handler module

Handler for keylist message.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_handler.KeylistHandler
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for keylist message.

async handle(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Handle keylist message.

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_query_handler module

Handler for keylist-query message.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_query_handler.KeylistQueryHandler
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for keylist-query message.

async handle(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Handle keylist-query message.

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_handler module

Handler for keylist-update messages.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_handler.KeylistUpdateHandler
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for keylist-update messages.

async handle(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Handle keylist-update messages.

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_response_handler module

Handler for keylist-update-response message.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_response_handler.KeylistUpdateResponseHandler
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for keylist-update-response message.

async handle(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Handle keylist-update-response message.

async notify_keylist_updated(*profile:* aries_cloudagent.core.profile.Profile, *connection_id: str*, *response:* aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update_response.K
Notify of keylist update response received.

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_deny_handler module

Handler for mediate-deny message.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_deny_handler.**MediationDenyHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for mediate-deny message.

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

Handle mediate-deny message.

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_grant_handler module

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_request_handler module

Handler for mediate-request message.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_request_handler.**MediationRequestHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for mediate-request message.

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

Handle mediate-request message.

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.problem_report_handler module

Coordinate mediation problem report message handler.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.problem_report_handler.**CMProblemReportHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for Coordinate Mediation Problem Report Message.

Updates the ConnRecord Metadata state.

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

Coordinate mediation problem report message handler.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages package

Subpackages

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner package

Submodules

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_key module

Inner structure of keylist message. Represents a single item in keylist.keys.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_key.**KeylistKey**(*, *recipient_key: Optional[st* = *None*, *action: Optional[st* = *None*, *result: Optional[st* = *None*, ***kwarg*

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

Inner structure of Keylist keys attribute.

class Meta

Bases: object

KeylistKey metadata.

schema_class = 'KeylistKeySchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_key.**KeylistKeySchema**(

Bases: marshmallow.

KeylistKey schema.

class Meta

Bases: object

KeylistKeySchema metadata.

model_class
alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_key.KeylistKey*

recipient_key

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_query_paginate module

Inner structure of KeylistQuery. Represents KeylistQuery.paginate.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_query_paginate.Keylis

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

Class representing a keylist query pagination.

class Meta
Bases: object

Keylist query pagination metadata.

schema_class = 'KeylistQueryPaginateSchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_query_paginate.Keylis

Bases: marshmallow.

Keylist query pagination schema.

class Meta
Bases: object

Keylist query pagination schema metadata.

model_class
alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_query_paginate.KeylistQueryPaginate*

limit

offset

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_update_rule module

Inner structure of keylist-update message.

Represents single item of keylist-update.updates.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_update_rule.**KeylistUp**

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

Class representing a keylist update rule.

class Meta

Bases: object

Keylist update metadata.

schema_class = 'KeylistUpdateRuleSchema'

RULE_ADD = 'add'

RULE_REMOVE = 'remove'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_update_rule.**KeylistUp**

Bases: marshmallow.

Keylist update specification schema.

class Meta

Bases: object

Keylist update schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_update_rule.KeylistUpdateRule*

action

recipient_key

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_updated module

Inner structure of keylist-update-response.

Represents single item in keylist-update-response.updated list.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_updated.KeylistUpdate

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

Class representing a route update response.

class Meta

Bases: object

KeylistUpdated metadata.

schema_class = 'KeylistUpdatedSchema'

RESULT_CLIENT_ERROR = 'client_error'

RESULT_NO_CHANGE = 'no_change'

RESULT_SERVER_ERROR = 'server_error'

RESULT_SUCCESS = 'success'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_updated.KeylistUpdate

Bases: marshmallow.

KeylistUpdated schema.

class Meta

Bases: object

KeylistUpdatedSchema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_updated.KeylistUpdated*

action

recipient_key

result

Submodules

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist module

Response to keylist-query message.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist.**Keylist**(*, *keys: Optional[Sequence[str]] = None*, *pagination: Optional[aries_cloudagent.p = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a keylist-query response.

class Meta

Bases: object

Metadata for a keylist query response.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_handler.KeylistHandler'

message_type = 'coordinate-mediation/1.0/keylist'

schema_class = 'KeylistSchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist.**KeylistSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Keylist query response schema class.

class Meta

Bases: object

Keylist query response schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist.Keylist*

keys

pagination

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_query module

keylist-query message used to request list of keys handled by mediator.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_query.**KeylistQuery**(*, *filter: Optional[dict* = *None*, *paginate: Optional[arie* = *None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a keylist query message.

class Meta

Bases: object

Metadata for a keylist query.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_query_handler.KeylistQueryHandler'

message_type = 'coordinate-mediation/1.0/keylist-query'

schema_class = 'KeylistQuerySchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_query.**KeylistQuerySchema**(*ar *An **k *An

Bases: marshmallow.

Keylist query schema class.

class Meta

Bases: object

Keylist query schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_query.KeylistQuery*

filter

paginate

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update module

keylist-update message used to notify mediator of keylist changes.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.**KeylistUpdate**(*, *updates: Optional[S = None*, ***kwarg*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a keylist update message.

class Meta

Bases: object

Metadata for a keylist update.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_handler.KeylistUpdateHandler'

message_type = 'coordinate-mediation/1.0/keylist-update'

schema_class = 'KeylistUpdateSchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.**KeylistUpdateSchema**(

Bases: marshmallow.

Keylist update schema class.

class Meta

Bases: object

Keylist update schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpdate*

updates

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update_response module

Response to keylist-update used to notify mediation client of applied updates.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update_response.KeylistUpda

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a keylist update result message.

class Meta

Bases: object

Metadata for a keylist update result.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_response_handler.KeylistUpdateResponseHandler'

message_type = 'coordinate-mediation/1.0/keylist-update-response'

schema_class = 'KeylistUpdateResponseSchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update_response.KeylistUpda

Bases: marshmallow.

Keylist update result schema class.

class Meta

Bases: object

Keylist update result schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update_response.KeylistUpdateResponse*

updated

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_deny module

mediate-deny message used to notify mediation client of a denied mediation request.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_deny.**MediationDeny**(*, *mediator_terms: Optional[Seq* = *None*, *recipient_terms: Optional[Seq* = *None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a mediation deny message.

class Meta

Bases: object

Metadata for a mediation deny.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_deny_handler.MediationDenyHandler'

message_type = 'coordinate-mediation/1.0/mediate-deny'

schema_class = 'MediationDenySchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_deny.**MediationDenySchema**(**a* *An* ***k* *An*

Bases: marshmallow.

Mediation grant schema class.

class Meta

Bases: object

Mediation deny schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_deny.MediationDeny*

mediator_terms

recipient_terms

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_grant module

mediate-grant message.

Used to notify mediation client of a granted mediation request.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_grant.**MediationGrant**(*, *endpoint: Optional[st* = *None*, *routing_keys Optional[S* = *None*, ***kwarg*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a mediation grant message.

class Meta

Bases: object

Metadata for a mediation grant.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_grant_handler.MediationGrantHandler'

message_type = 'coordinate-mediation/1.0/mediate-grant'

schema_class = 'MediationGrantSchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_grant.**MediationGrantSchema**(

Bases: marshmallow.

Mediation grant schema class.

class Meta

Bases: object

Mediation grant schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_grant.MediationGrant*

endpoint

routing_keys

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_request module

mediate-request message used to request mediation from a mediator.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_request.**MediationRequest**(*, *mediator_terms: Optional[Sequence[str]] = None*, *recipient_terms: Optional[Sequence[str]] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Represents a request for mediation.

class Meta

Bases: object

MediationRequest metadata.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_request_handler.MediationRequestHandler'

message_type = 'coordinate-mediation/1.0/mediate-request'

schema_class = 'MediationRequestSchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_request.**MediationRequestSchema**(*, *only: Optional[Sequence[str]] = None*, *exclude: Sequence[str] = ()*, *many: bool = False*, *context: Optional[dict] = None*, *load_only: Sequence[str] = ()*, *dump_only: Sequence[str] = ()*, *partial: Optional[bool] = False*, *unknown: Optional[str] = None*)

Bases: marshmallow.

Mediation request schema class.

class Meta

Bases: object

Mediation request schema metadata.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_request.MediationRequest*

mediator_terms

recipient_terms

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.problem_report module

Represents a coordinate-mediation problem report message.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.problem_report.**CMProblemReport**(**args*, ***kwa*

Bases: *aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReport*

Base class representing a coordinate mediation problem report message.

class Meta

Bases: object

CMProblemReport metadata.

handler_class = 'aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.problem_report_handler.CMProblemReportHandler'

message_type = 'coordinate-mediation/1.0/problem-report'

schema_class = 'CMProblemReportSchema'

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.problem_report.**CMProblemReportSchema**

Bases: marshmallow.

Schema for ProblemReport base class.

class Meta

Bases: object

Metadata for problem report schema.

model_class

alias of *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.problem_report.CMProblemReport*

validate_fields(*data*, ***kwargs*)

Validate schema fields.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.problem_report.**ProblemReportReason**(

Bases: enum.Enum

Supported reason codes.

MEDIATION_NOT_GRANTED = 'mediation_not_granted'

MEDIATION_REQUEST_REPEAT = 'request_already_exists_from_connection'

aries_cloudagent.protocols.coordinate_mediation.v1_0.models package

Submodules

aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record module

Store state for Mediation requests.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.**MediationRecord**(*, *mediation_id: Optional = None*, *state: Optional = None*, *role: Optional = None*, *connection_id: Optional = None*, *mediator_terms: Optional = None*, *recipient_terms: Optional = None*, *routing_keys: Optional = None*, *endpoint: Optional = None*, ***kwa*

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

Class representing stored mediation information.

class Meta

Bases: object

RouteRecord metadata.

schema_class = 'MediationRecordSchema'

RECORD_ID_NAME = 'mediation_id'

RECORD_TOPIC: Optional[str] = 'mediation'

RECORD_TYPE = 'mediation_requests'

ROLE_CLIENT = 'client'

ROLE_SERVER = 'server'

STATE_DENIED = 'denied'

STATE_GRANTED = 'granted'

STATE_REQUEST = 'request'

TAG_NAMES = {'connection_id', 'role', 'state'}

async classmethod exists_for_connection_id(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: str*) → bool

Return whether a mediation record exists for the given connection.

Parameters

- **session** (*ProfileSession*) – session
- **connection_id** (*str*) – connection_id

Returns whether record exists

Return type bool

property mediation_id: str

Get Mediation ID.

property record_value: dict

Return values of record as dictionary.

async classmethod retrieve_by_connection_id(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: str*) → *aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediati*

Retrieve a mediation record by connection ID.

Parameters

- **session** (*ProfileSession*) – session
- **connection_id** (*str*) – connection_id

Returns retrieved record

Return type *MediationRecord*

property state: str

Get Mediation state.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecordSchema

Bases: marshmallow.

MediationRecordSchema schema.

class Meta
Bases: object

MediationRecordSchema metadata.

model_class
alias of aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord

connection_id

endpoint

mediation_id

mediator_terms

recipient_terms

role

routing_keys

Submodules

aries_cloudagent.protocols.coordinate_mediation.v1_0.controller module

Protocol controller for coordinate mediation.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.controller.Controller(*protocol: str*)

Bases: object

Coordinate mediation protocol controller.

determine_goal_codes() → Sequence[str]
Return defined goal_codes.

aries_cloudagent.protocols.coordinate_mediation.v1_0.manager module

Manager for Mediation coordination.

exception aries_cloudagent.protocols.coordinate_mediation.v1_0.manager.MediationAlreadyExists(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: aries_cloudagent.protocols.coordinate_mediation.v1_0.manager.

MediationManagerError

Raised on mediation record already exists for given connection.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.manager.**MediationManager**(*profile:* aries_cloudagent.core.pro

Bases: object

Class for handling Mediation.

MediationManager creates or retrieves a routing DID as a means to hand out a consistent routing key to mediation clients.

DEFAULT_MEDIATOR_RECORD_TYPE = 'default_mediator'

KEYLIST_UPDATED_EVENT = 'acapy::keylist::updated'

METADATA_ID = 'id'

METADATA_KEY = 'mediation'

ROUTING_DID_RECORD_TYPE = 'routing_did'

SEND_REQ_AFTER_CONNECTION = 'send_mediation_request_on_connection'

SET_TO_DEFAULT_ON_GRANTED = 'set_to_default_on_granted'

async add_key(*recipient_key: str*, *message: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpdate*] = None*) → *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpdate*

Prepare a keylist update add.

Parameters

- **recipient_key** (*str*) – key to add
- **message** (*Optional[KeylistUpdate]*) – append update to message

Returns Message to send to mediator to notify of key addition.

Return type KeylistUpdate

async clear_default_mediator()

Clear the stored default mediator.

async create_keylist_query_response(*keylist: Sequence[*aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecor → *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist.Keylist*

Prepare a keylist message from keylist.

Parameters **keylist** (*Sequence[RouteRecord]*) – keylist to format into message

Returns message to return to client

Return type Keylist

async deny_request(*mediation_id: str*, ***, *mediator_terms: Optional[Sequence[str]] = None*, *recipient_terms: Optional[Sequence[str]] = None*) → Tuple[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord*, *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_deny.MediationDeny*]

Deny a mediation request and prepare a deny message.

Parameters

- **mediation_id** – mediation record ID to deny
- **mediator_terms** (*Sequence[str]*) – updated mediator terms to return to
- **requester.** –
- **recipient_terms** (*Sequence[str]*) – updated recipient terms to return to
- **requester.** –

Returns message to return to denied client.

Return type MediationDeny

async get_default_mediator() → Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.Medi*

Retrieve default mediator from the store.

Returns retrieved default mediator or None if not set

Return type Optional[MediationRecord]

async get_default_mediator_id() → Optional[str]

Retrieve default mediator ID from the store.

Returns retrieved default mediator ID or None if not set

Return type Optional[str]

async get_keylist(*record:* aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord) → Sequence[*aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord*]

Retrieve keylist for mediation client.

Parameters **record** (*MediationRecord*) – record associated with client keylist

Returns sequence of routes (the keylist)

Return type Sequence[RouteRecord]

async get_my_keylist(*connection_id: Optional[str] = None*) → Sequence[*aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord*]

Get my routed keys.

Parameters **connection_id** (*Optional[str]*) – connection id of mediator

Returns list of routes (the keylist)

Return type Sequence[RouteRecord]

async grant_request(*mediation_id: str*) → Tuple[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord*, *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_grant.MediationGrant*]

Grant a mediation request and prepare grant message.

Parameters **mediation_id** – mediation record ID to grant

Returns updated mediation record and message to return to grantee

Return type (MediationRecord, MediationGrant)

async prepare_keylist_query(*filter_: Optional[dict] = None*, *paginate_limit: int = - 1*, *paginate_offset: int = 0*) → *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_query.KeylistQuery*

Prepare keylist query message.

Parameters

- **filter** (*dict*) – filter for keylist query
- **paginate_limit** (*int*) – paginate_limit
- **paginate_offset** (*int*) – paginate_offset

Returns message to send to mediator

Return type KeylistQuery

async prepare_request(*connection_id: str*, *mediator_terms: Optional[Sequence[str]] = None*, *recipient_terms: Optional[Sequence[str]] = None*) → Tuple[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRec*, *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_request.MediationReques*]

Prepare a MediationRequest Message, saving a new mediation record.

Parameters

- **connection_id** (*str*) – ID representing mediator
- **mediator_terms** (*Sequence[str]*) – mediator_terms
- **recipient_terms** (*Sequence[str]*) – recipient_terms

Returns message to send to mediator

Return type MediationRequest

async receive_request(*connection_id: str*, *request:* aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_request.MediationRequest → *aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord*

Create a new mediation record to track this request.

Parameters **request** (*MediationRequest*) – request message

Returns record created during receipt of request.

Return type MediationRecord

async remove_key(*recipient_key: str*, *message: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpdate*] = None*) → *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpdate*

Prepare keylist update remove.

Parameters

- **recipient_key** (*str*) – key to remove
- **message** (*Optional[KeylistUpdate]*) – append update to message

Returns Message to send to mediator to notify of key removal.

Return type KeylistUpdate

async request_denied(*record:* aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord, *deny:* aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_deny.MediationDeny)

Process mediation denied message.

Parameters **record** (*MediationRecord*) – record representing the denied mediation request

async request_granted(*record:* aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord, *grant:* aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_grant.MediationGrant)

Process mediation grant message.

Parameters **record** (*MediationRecord*) – record representing the granted mediation request

async set_default_mediator(*record:* aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationR

Set default mediator from record.

async set_default_mediator_by_id(*mediation_id: str*)

Set default mediator from ID.

async store_update_results(*connection_id: str*, *results: Sequence[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_update

Store results of keylist update from keylist update response message.

Parameters

- **connection_id** (*str*) – connection ID of mediator sending results
- **results** (*Sequence[KeylistUpdated]*) – keylist update results
- **session** – An active profile session

async update_keylist(*record:* aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecord, *updates: Sequence[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_update_rule.Ke → *aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update_response.KeylistUpd*

Update routes defined in keylist update rules.

Parameters

- **record** (*MediationRecord*) – record associated with client updating keylist
- **updates** (*Sequence[KeylistUpdateRule]*) – updates to apply

Returns message to return to client

Return type KeylistUpdateResponse

exception aries_cloudagent.protocols.coordinate_mediation.v1_0.manager.**MediationManagerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Generic Mediation error.

exception aries_cloudagent.protocols.coordinate_mediation.v1_0.manager.**MediationNotGrantedError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.protocols.coordinate_mediation.v1_0.manager.MediationManagerError*

Raised when mediation state should be granted and is not.

aries_cloudagent.protocols.coordinate_mediation.v1_0.message_types module

Message type identifiers for Coordinate Mediation protocol.

aries_cloudagent.protocols.coordinate_mediation.v1_0.normalization module

Normalization methods used while transitioning to DID:Key method.

aries_cloudagent.protocols.coordinate_mediation.v1_0.normalization.**normalize_from_did_key**(*key: str*)

Normalize Recipient/Routing keys from DID:Key to public keys.

aries_cloudagent.protocols.coordinate_mediation.v1_0.normalization.**normalize_from_public_key**(*key: str*)

Normalize Recipient/Routing keys from public keys to DID:Key.

aries_cloudagent.protocols.coordinate_mediation.v1_0.route_manager module

Route manager.

Set up routing for newly formed connections.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.route_manager.**CoordinateMediationV1RouteManager**

Bases: *aries_cloudagent.protocols.coordinate_mediation.v1_0.route_manager.RouteManager*

Manage routes using Coordinate Mediation protocol.

async **routing_info**(*profile:* aries_cloudagent.core.profile.Profile, *my_endpoint: str*, *mediation_record: Optional[aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecor = None*) → Tuple[List[str], str]

Return routing info for mediator.

class aries_cloudagent.protocols.coordinate_mediation.v1_0.route_manager.**RouteManager**

Bases: abc.ABC

Base Route Manager.

async **connection_from_recipient_key**(*profile:* aries_cloudagent.core.profile.Profile, *recipient_key: str*) → *aries_cloudagent.connections.models.conn_record.ConnRecord*

Retrieve connection for a recipient_key.

The recipient key is expected to be a local key owned by this agent.

async get_or_create_my_did(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord) → *aries_cloudagent.wallet.did_info.DIDInfo*

Create or retrieve DID info for a conneciton.

async mediation_record_for_connection(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *mediation_id: Optional[str] = None*, *or_default: bool = False*)

Return relevant mediator for connection.

async mediation_record_if_id(*profile:* aries_cloudagent.core.profile.Profile, *mediation_id: Optional[str] = None*, *or_default: bool = False*)

Validate mediation and return record.

If mediation_id is not None, validate mediation record state and return record else, return None

async route_connection(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *mediation_record: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.Mediationl *= None*) → Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpda*

Set up routing for a connection.

This method will evaluate connection state and call the appropriate methods.

async route_connection_as_invitee(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *mediation_record: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_rec *= None*) → Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_upda*

Set up routing for a new connection when we are the invitee.

async route_connection_as_inviter(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *mediation_record: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_rec *= None*) → Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_upda*

Set up routing for a new connection when we are the inviter.

async route_invitation(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *mediation_record: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.Mediationl *= None*) → Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpda*

Set up routing for receiving a response to an invitation.

async route_public_did(*profile:* aries_cloudagent.core.profile.Profile, *verkey: str*)

Establish routing for a public DID.

async route_static(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *mediation_record: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.MediationRecor = *None*) → Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update.KeylistUpdate*]

Establish routing for a static connection.

abstract async routing_info(*profile:* aries_cloudagent.core.profile.Profile, *my_endpoint: str*, *mediation_record: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record.Med = *None*) → Tuple[List[str], str]

Retrieve routing keys.

async save_mediator_for_connection(*profile:* aries_cloudagent.core.profile.Profile, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *mediation_record: Optional[*aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_re = *None*, *mediation_id: Optional[str] = None*)

Save mediator info to connection metadata.

exception aries_cloudagent.protocols.coordinate_mediation.v1_0.route_manager.RouteManagerError

Bases: Exception

Raised on error from route manager.

aries_cloudagent.protocols.coordinate_mediation.v1_0.route_manager_provider module

aries_cloudagent.protocols.coordinate_mediation.v1_0.routes module

Submodules

aries_cloudagent.protocols.coordinate_mediation.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store module

Storage management for configuration-provided mediation invite.

Handle storage and retrieval of mediation invites provided through arguments. Enables having the mediation invite config be the same for *provision* and *starting* commands.

class aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.MediationInviteRecord(*invite str*, *used: bool*)

Bases: tuple

A record to store mediation invites and their freshness.

static from_json(*json_invite_record: str*) → *aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.MediationInviteRecord*

Returns a mediation invite record deserialized from a json string.

property invite
Alias for field number 0

to_json() → str

Returns The current record serialized into a json string.

static unused(*invite: str*) → *aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.MediationInviteRecord*

Parameters **invite** – invite string as provided by the mediator.

Returns An unused mediation invitation for the given invite string

property used
Alias for field number 1

class aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.**MediationInviteStore**(*storage*
aries_c

Bases: object

Store and retrieve mediation invite configuration.

INVITE_RECORD_CATEGORY = 'config'

MEDIATION_INVITE_ID = 'mediation_invite'

async get_mediation_invite_record(*provided_mediation_invitation: Optional[str]*) → Optional[*aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.Med*

Provide the MediationInviteRecord to use/that was used for mediation.

Returned record may have been used already.

Stored record is updated if *provided_mediation_invitation* has changed. Updated record is marked as unused.

Parameters **provided_mediation_invitation** – mediation invite provided by user

Returns mediation invite to use/that was used to connect to the mediator. None if no invitation was provided/provisioned.

async mark_default_invite_as_used()
Mark the currently stored invitation as used if one exists.

Raises ***NoDefaultMediationInviteException*** – if trying to mark invite as used when there is no invite stored.

async store(*mediation_invite:* aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.MediationInviteRecord) → *aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.MediationInviteRecord*

Store the mediator's invite for further use when starting the agent.

Update the currently stored invite if one already exists. This assumes a new invite and as such, marks it as unused.

Parameters **mediation_invite** – mediation invite url

Returns stored mediation invite

exception aries_cloudagent.protocols.coordinate_mediation.mediation_invite_store.NoDefaultMediationInviteException

Bases: Exception

Raised if trying to mark a default invite as used when none exist.

aries_cloudagent.protocols.didexchange package

Subpackages

aries_cloudagent.protocols.didexchange.v1_0 package

Subpackages

aries_cloudagent.protocols.didexchange.v1_0.handlers package

Submodules

aries_cloudagent.protocols.didexchange.v1_0.handlers.complete_handler module

aries_cloudagent.protocols.didexchange.v1_0.handlers.invitation_handler module

Connect invitation handler under RFC 23 (DID exchange).

class aries_cloudagent.protocols.didexchange.v1_0.handlers.invitation_handler.InvitationHandler

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for connection invitation message under RFC 23 (DID exchange).

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

Handle connection invitation under RFC 23 (DID exchange).

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.didexchange.v1_0.handlers.request_handler module

aries_cloudagent.protocols.didexchange.v1_0.handlers.response_handler module

aries_cloudagent.protocols.didexchange.v1_0.messages package

Submodules

aries_cloudagent.protocols.didexchange.v1_0.messages.complete module

Represents a DID exchange complete message under RFC 23.

class aries_cloudagent.protocols.didexchange.v1_0.messages.complete.**DIDXComplete**(***kwargs*)
Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a DID exchange completion.

class Meta
Bases: object

Metadata for DID exchange completion.

handler_class = 'aries_cloudagent.protocols.didexchange.v1_0.handlers.complete_handler.DIDXCompleteHandler'

message_type = 'didexchange/1.0/complete'

schema_class = 'DIDXCompleteSchema'

class aries_cloudagent.protocols.didexchange.v1_0.messages.complete.**DIDXCompleteSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

DID exchange complete schema class.

class Meta
Bases: object

DID exchange complete schema metadata.

model_class
alias of *aries_cloudagent.protocols.didexchange.v1_0.messages.complete.DIDXComplete*

check_thread_deco(*obj*, ***kwargs*)
Thread decorator, and its thid and pthid, are mandatory.

aries_cloudagent.protocols.didexchange.v1_0.messages.problem_report_reason module

DID Exchange problem report reasons.

class aries_cloudagent.protocols.didexchange.v1_0.messages.problem_report_reason.**ProblemReportReason**(*val*
Bases: enum.Enum

Supported reason codes.

COMPLETE_NOT_ACCEPTED = 'complete_not_accepted'

INVITATION_NOT_ACCEPTED = 'invitation_not_accepted'

REQUEST_NOT_ACCEPTED = 'request_not_accepted'

REQUEST_PROCESSING_ERROR = 'request_processing_error'

RESPONSE_NOT_ACCEPTED = 'response_not_accepted'

RESPONSE_PROCESSING_ERROR = 'response_processing_error'

aries_cloudagent.protocols.didexchange.v1_0.messages.request module

Represents a DID exchange request message under RFC 23.

class aries_cloudagent.protocols.didexchange.v1_0.messages.request.**DIDXRequest**(*, *label: Optional[str] = None*, *did: Optional[str] = None*, *did_doc_attach: Optional[aries_cloudagent.messag = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a DID exchange request under RFC 23.

class Meta

Bases: object

Metadata for DID exchange request under RFC 23.

handler_class = 'aries_cloudagent.protocols.didexchange.v1_0.handlers.request_handler.DIDXRequestHandler'

message_type = 'didexchange/1.0/request'

schema_class = 'DIDXRequestSchema'

class aries_cloudagent.protocols.didexchange.v1_0.messages.request.**DIDXRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema class for DID exchange request under RFC 23.

class Meta

Bases: object

DID exchange request schema class metadata.

model_class

alias of *aries_cloudagent.protocols.didexchange.v1_0.messages.request.DIDXRequest*

did_doc_attach

alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorSchema*

aries_cloudagent.protocols.didexchange.v1_0.messages.response module

Represents a DID exchange response message under RFC 23.

class aries_cloudagent.protocols.didexchange.v1_0.messages.response.**DIDXResponse**(*, *did: Optional[str] = None*, *did_doc_attach: Optional[*aries_cloudagent.mess *= None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a DID exchange response under RFC 23.

class Meta

Bases: object

Metadata for DID exchange response under RFC 23.

handler_class = 'aries_cloudagent.protocols.didexchange.v1_0.handlers.response_handler.DIDXResponseHandler'

message_type = 'didexchange/1.0/response'

schema_class = 'DIDXResponseSchema'

class aries_cloudagent.protocols.didexchange.v1_0.messages.response.**DIDXResponseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema class for DID exchange response under RFC 23.

class Meta

Bases: object

DID exchange response schema class metadata.

model_class

alias of *aries_cloudagent.protocols.didexchange.v1_0.messages.response.DIDXResponse*

did

did_doc_attach

Submodules

aries_cloudagent.protocols.didexchange.v1_0.manager module

aries_cloudagent.protocols.didexchange.v1_0.message_types module

Message type identifiers for Connections.

aries_cloudagent.protocols.didexchange.v1_0.routes module

Submodules

aries_cloudagent.protocols.didexchange.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.discovery package

Subpackages

aries_cloudagent.protocols.discovery.v1_0 package

Subpackages

aries_cloudagent.protocols.discovery.v1_0.handlers package

Submodules

aries_cloudagent.protocols.discovery.v1_0.handlers.disclose_handler module

Handler for incoming disclose messages.

class aries_cloudagent.protocols.discovery.v1_0.handlers.disclose_handler.**DiscloseHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming disclose messages.

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

Message handler implementation.

aries_cloudagent.protocols.discovery.v1_0.handlers.query_handler module

Handler for incoming query messages.

class aries_cloudagent.protocols.discovery.v1_0.handlers.query_handler.**QueryHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming query messages.

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

Message handler implementation.

aries_cloudagent.protocols.discovery.v1_0.messages package

Submodules

aries_cloudagent.protocols.discovery.v1_0.messages.disclose module

Represents a feature discovery disclosure message.

class aries_cloudagent.protocols.discovery.v1_0.messages.disclose.**Disclose**(*, *protocols: Optional[Sequence[Mapping[str, Mapping]]] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Represents a feature discovery disclosure, the response to a query message.

class Meta

Bases: object

Disclose metadata.

handler_class = 'aries_cloudagent.protocols.discovery.v1_0.handlers.disclose_handler.DiscloseHandler'

message_type = 'discover-features/1.0/disclose'

schema_class = 'DiscloseSchema'

class aries_cloudagent.protocols.discovery.v1_0.messages.disclose.**DiscloseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Disclose message schema used in serialization/deserialization.

class Meta

Bases: object

DiscloseSchema metadata.

model_class

alias of *aries_cloudagent.protocols.discovery.v1_0.messages.disclose.Disclose*

protocols

class aries_cloudagent.protocols.discovery.v1_0.messages.disclose.**ProtocolDescriptorSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for an entry in the protocols list.

pid

roles

aries_cloudagent.protocols.discovery.v1_0.messages.query module

Represents a feature discovery query message.

class aries_cloudagent.protocols.discovery.v1_0.messages.query.**Query**(***, *query: Optional[str] = None*, *comment: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Represents a feature discovery query.

Used for inspecting what message types are supported by the agent.

class Meta

Bases: object

Query metadata.

handler_class = 'aries_cloudagent.protocols.discovery.v1_0.handlers.query_handler.QueryHandler'

message_type = 'discover-features/1.0/query'

schema_class = 'QuerySchema'

class aries_cloudagent.protocols.discovery.v1_0.messages.query.**QuerySchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query message schema used in serialization/deserialization.

class Meta

Bases: object

QuerySchema metadata.

model_class

alias of *aries_cloudagent.protocols.discovery.v1_0.messages.query.Query*

comment

query

aries_cloudagent.protocols.discovery.v1_0.models package

Package-wide code and data.

Submodules

aries_cloudagent.protocols.discovery.v1_0.models.discovery_record module

.

class aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.**V10DiscoveryExchangeRecord**(*, *discovery_e* *Optional* = *None*, *connection_i* *Optional* = *None*, *thread* *Optional* = *None*, *query* *Optional* aries_ = *None*, *disclose* *Optional* aries_ = *None*, ***kwa*

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

Represents a Discover Feature (0031) exchange record.

class Meta

Bases: object

V10DiscoveryExchangeRecord metadata.

schema_class = 'V10DiscoveryRecordSchema'

RECORD_ID_NAME = 'discovery_exchange_id'

RECORD_TOPIC: Optional[str] = 'discover_feature'

RECORD_TYPE = 'discovery_exchange_v10'

TAG_NAMES = {'connection_id', 'thread_id'}

property disclose:
aries_cloudagent.protocols.discovery.v1_0.messages.disclose.Disclose
Accessor; get deserialized view.

property discovery_exchange_id: str
Accessor for the ID.

async classmethod exists_for_connection_id(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: str*) → bool
Return whether a discovery exchange record exists for the given connection.

Parameters

- **session** (*ProfileSession*) – session
- **connection_id** (*str*) – connection_id

Returns whether record exists

Return type bool

property query_msg: *aries_cloudagent.protocols.discovery.v1_0.messages.query.Query*
Accessor; get deserialized view.

property record_value: dict
Accessor for the JSON record value generated.

async classmethod retrieve_by_connection_id(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: str*) → *aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.V1*
Retrieve a discovery exchange record by connection.

class aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.**V10DiscoveryRecordSchema**(**args: Any*, ***kwarg Any*)

Bases: marshmallow.

Schema to allow ser/deser of Discover Feature (0031) records.

class Meta
Bases: object

V10DiscoveryRecordSchema metadata.

model_class
alias of *aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.V10DiscoveryExchangeRecord*

connection_id

disclose

discovery_exchange_id

query_msg

thread_id

Submodules

aries_cloudagent.protocols.discovery.v1_0.manager module

Classes to manage discover features.

class aries_cloudagent.protocols.discovery.v1_0.manager.**V10DiscoveryMgr**(*profile:* aries_cloudagent.core.profile.Profile)

Bases: object

Class for discover feature v1_0 under RFC 31.

async check_if_disclosure_received(*record_id: str*) → *aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.V10Discovery*

Check if disclosures has been received.

async create_and_send_query(*query: str*, *comment: Optional[str] = None*, *connection_id: Optional[str] = None*) → *aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.V10DiscoveryExchange*

Create and send a Query message.

async lookup_exchange_rec_by_connection(*connection_id: str*) → Optional[*aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.V*

Retrieve V20DiscoveryExchangeRecord by connection_id.

property profile: *aries_cloudagent.core.profile.Profile*

Accessor for the current Profile.

Returns The Profile for this manager

async receive_disclose(*disclose_msg:* aries_cloudagent.protocols.discovery.v1_0.messages.disclose.Disclose, *connection_id: str*) → *aries_cloudagent.protocols.discovery.v1_0.models.discovery_record.V10DiscoveryExchangeRecor*

Receive Disclose message and return updated V10DiscoveryExchangeRecord.

async receive_query(*query_msg:* aries_cloudagent.protocols.discovery.v1_0.messages.query.Query) → *aries_cloudagent.protocols.discovery.v1_0.messages.disclose.Disclose*

Process query and return the corresponding disclose message.

exception aries_cloudagent.protocols.discovery.v1_0.manager.**V10DiscoveryMgrError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Discover feature v1_0 error.

aries_cloudagent.protocols.discovery.v1_0.message_types module

Message type identifiers for Feature Discovery.

aries_cloudagent.protocols.discovery.v1_0.routes module

Feature discovery admin routes.

class aries_cloudagent.protocols.discovery.v1_0.routes.**QueryDiscoveryExchRecordsSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query string parameter for Discover Features v1.0 exchange record.

connection_id

class aries_cloudagent.protocols.discovery.v1_0.routes.**QueryFeaturesQueryStringSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query string parameters for feature query.

comment

connection_id

query

class aries_cloudagent.protocols.discovery.v1_0.routes.**V10DiscoveryExchangeListResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for Discover Features v1.0 exchange records.

results

aries_cloudagent.protocols.discovery.v1_0.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.discovery.v1_0.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

aries_cloudagent.protocols.discovery.v2_0 package

Subpackages

aries_cloudagent.protocols.discovery.v2_0.handlers package

Submodules

aries_cloudagent.protocols.discovery.v2_0.handlers.disclosures_handler module

Handler for incoming disclose messages.

class aries_cloudagent.protocols.discovery.v2_0.handlers.disclosures_handler.**DisclosuresHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming disclose messages.

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

Message handler implementation.

aries_cloudagent.protocols.discovery.v2_0.handlers.queries_handler module

Handler for incoming queries messages.

class aries_cloudagent.protocols.discovery.v2_0.handlers.queries_handler.**QueriesHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming queries messages.

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

Message handler implementation.

aries_cloudagent.protocols.discovery.v2_0.messages package

Submodules

aries_cloudagent.protocols.discovery.v2_0.messages.disclosures module

Represents a feature discovery disclosure message.

class aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.**Disclosures**(*, *disclosures: Optional[Sequence[Mapping]] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Represents a feature discovery disclosure, the response to a query message.

class Meta
Bases: object

Disclose metadata.

handler_class = 'aries_cloudagent.protocols.discovery.v2_0.handlers.disclosures_handler.DisclosuresHandler'

message_type = 'discover-features/2.0/disclosures'

schema_class = 'DisclosuresSchema'

class aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.DisclosuresSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Disclose message schema used in serialization/deserialization.

class Meta
Bases: object

DiscloseSchema metadata.

model_class
alias of aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.Disclosures

disclosures

class aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.GoalCodeDescriptorSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Schema for an entry in the goal_code list.

feature_type

id

class aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.ProtocolDescriptorSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Schema for an entry in the protocols list.

feature_type

id

roles

class aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.ProtocolOrGoalCodeDescriptorField(

Bases: marshmallow.fields.

ProtocolDescriptor or GoalCodeDescriptor for Marshmallow.

aries_cloudagent.protocols.discovery.v2_0.messages.queries module

Represents a feature discovery queries message.

class aries_cloudagent.protocols.discovery.v2_0.messages.queries.**Queries**(*, *queries: Optional[Sequence[*aries_cloudagent.protoc = *None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Represents a discover-features v2 queries message.

Used for inspecting what message types are supported by the agent.

class Meta

Bases: object

Queries metadata.

handler_class = 'aries_cloudagent.protocols.discovery.v2_0.handlers.queries_handler.QueriesHandler'

message_type = 'discover-features/2.0/queries'

schema_class = 'QueriesSchema'

class aries_cloudagent.protocols.discovery.v2_0.messages.queries.**QueriesSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query message schema used in serialization/deserialization.

class Meta

Bases: object

QuerySchema metadata.

model_class

alias of *aries_cloudagent.protocols.discovery.v2_0.messages.queries.Queries*

queries

class aries_cloudagent.protocols.discovery.v2_0.messages.queries.**QueryItem**(*, *feature_type: str*, *match: str*)

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

Defines QueryItem field.

class Meta

Bases: object

QueryItem metadata.

schema_class = 'QueryItemSchema'

class aries_cloudagent.protocols.discovery.v2_0.messages.queries.**QueryItemSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single QueryItem Schema.

class Meta
Bases: object

QueryItemSchema metadata.

model_class
alias of aries_cloudagent.protocols.discovery.v2_0.messages.queries.QueryItem

feature_type

match

aries_cloudagent.protocols.discovery.v2_0.models package

Package-wide code and data.

Submodules

aries_cloudagent.protocols.discovery.v2_0.models.discovery_record module

.

class aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.**V20DiscoveryExchangeRecord**(*, *discovery_e* *Optional* = *None*, *connection_i* *Optional* = *None*, *thread* *Optional* = *None*, *queri* *Optional* aries_ = *None*, *disclosures.* *Optional* aries_ = *None*, ***kwa*

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

Represents a Discover Feature v2_0 (0557) exchange record.

class Meta

Bases: object

V20DiscoveryExchangeRecord metadata.

schema_class = 'V20DiscoveryRecordSchema'

RECORD_ID_NAME = 'discovery_exchange_id'

RECORD_TOPIC: Optional[str] = 'discover_feature_v2_0'

RECORD_TYPE = 'discovery_exchange_v20'

TAG_NAMES = {'connection_id', 'thread_id'}

property disclosures:
aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.Disclosures

Accessor; get deserialized view.

property discovery_exchange_id: str
Accessor for the ID.

async classmethod exists_for_connection_id(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: str*) → bool
Return whether a discovery exchange record exists for the given connection.

Parameters

- **session** (*ProfileSession*) – session
- **connection_id** (*str*) – connection_id

Returns whether record exists

Return type bool

property queries_msg: *aries_cloudagent.protocols.discovery.v2_0.messages.queries.Queries*
Accessor; get deserialized view.

property record_value: dict
Accessor for the JSON record value generated.

async classmethod retrieve_by_connection_id(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: str*) → *aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.V2*
Retrieve a discovery exchange record by connection.

class aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.**V20DiscoveryRecordSchema**(**args: Any*, ***kwarg Any*)

Bases: marshmallow.

Schema to allow ser/deser of Discover Feature v2_0 records.

class Meta
Bases: object

V20DiscoveryRecordSchema metadata.

model_class
alias of *aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.V20DiscoveryExchangeRecord*

connection_id

disclosures

discovery_exchange_id

queries_msg

thread_id

Submodules

aries_cloudagent.protocols.discovery.v2_0.manager module

Classes to manage discover features.

class aries_cloudagent.protocols.discovery.v2_0.manager.**V20DiscoveryMgr**(*profile:* aries_cloudagent.core.profile.Profile)

Bases: object

Class for discover feature v1_0 under RFC 31.

async check_if_disclosure_received(*record_id: str*) → *aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.V20Discovery*

Check if disclosures has been received.

async create_and_send_query(*connection_id: Optional[str] = None*, *query_protocol: Optional[str] = None*, *query_goal_code: Optional[str] = None*) → *aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.V20DiscoveryExchange*

Create and send a Query message.

async execute_goal_code_query(*match: str*)

Execute goal code match query.

async execute_protocol_query(*match: str*)

Execute protocol match query.

async lookup_exchange_rec_by_connection(*connection_id: str*) → Optional[*aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.V*

Retrieve V20DiscoveryExchangeRecord by connection_id.

async proactive_disclose_features(*connection_id: str*)

Proactively dislose features on active connection setup.

property profile: *aries_cloudagent.core.profile.Profile*

Accessor for the current Profile.

Returns The Profile for this manager

async receive_disclose(*disclose_msg:* aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.Disclosures, *connection_id: Optional[str] = None*) → *aries_cloudagent.protocols.discovery.v2_0.models.discovery_record.V20DiscoveryExchangeRecor*

Receive Disclose message and return updated V20DiscoveryExchangeRecord.

async receive_query(*queries_msg:* aries_cloudagent.protocols.discovery.v2_0.messages.queries.Queries) → *aries_cloudagent.protocols.discovery.v2_0.messages.disclosures.Disclosures*

Process query and return the corresponding disclose message.

async return_to_publish_features() → Tuple[Optional[Sequence[str]], Optional[Sequence[str]]]

Return to_publish features filter, if specified.

exception aries_cloudagent.protocols.discovery.v2_0.manager.**V20DiscoveryMgrError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Discover feature v2_0 error.

aries_cloudagent.protocols.discovery.v2_0.message_types module

Message type identifiers for Feature Discovery.

aries_cloudagent.protocols.discovery.v2_0.routes module

Feature discovery v2 admin routes.

class aries_cloudagent.protocols.discovery.v2_0.routes.**QueryDiscoveryExchRecordsSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query string parameter for Discover Features v2.0 exchange record.

connection_id

class aries_cloudagent.protocols.discovery.v2_0.routes.**QueryFeaturesQueryStringSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query string parameters for feature query.

connection_id

query_goal_code

query_protocol

class aries_cloudagent.protocols.discovery.v2_0.routes.**V20DiscoveryExchangeListResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for Discover Features v2.0 exchange records.

results

class aries_cloudagent.protocols.discovery.v2_0.routes.**V20DiscoveryExchangeResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for Discover Features v2.0 exchange record.

results

aries_cloudagent.protocols.discovery.v2_0.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.discovery.v2_0.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

Submodules

aries_cloudagent.protocols.discovery.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.endorse_transaction package

Subpackages

aries_cloudagent.protocols.endorse_transaction.v1_0 package

Subpackages

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers package

Submodules

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.endorsed_transaction_response_handler module

Endorsed transaction response handler.

class aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.endorsed_transaction_response_handler.**EndorsedTransactionResponseHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for Endorsed transaction response.

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

Handle endorsed transaction response.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.refused_transaction_response_handler module

Refused transaction response handler.

class aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.refused_transaction_response_handler.**RefusedTransactionResponseHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for Refused transaction response.

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

Handle refused transaction response.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_acknowledgement_handler module

Transaction acknowledgement message handler.

class aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_acknowledgement_handler.**TransactionAcknowledgementHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for Acknowledging transaction.

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

Handle transaction acknowledgement message.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_cancel_handler module

Cancel transaction request handler.

class aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_cancel_handler.**TransactionCancelHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for Cancel transaction request.

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

Handle cancel transaction request.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_job_to_send_handler module

Transaction Job to send handler.

class aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_job_to_send_handler.**TransactionJobToSendHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for sending transaction jobs.

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

Handle transaction jobs.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_request_handler module

Transaction request handler.

class aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_request_handler.**TransactionRequestHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for transaction request.

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

Handle transaction request.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_resend_handler module

Transaction resend handler.

class aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_resend_handler.**TransactionResendHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler class for transaction resend.

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

Handle transaction resend.

Parameters

- **context** – Request context
- **responder** – Responder callback

aries_cloudagent.protocols.endorse_transaction.v1_0.messages package

Submodules

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.cancel_transaction module

Represents a cancel transaction message.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.cancel_transaction.**CancelTransaction**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a cancel transaction message.

class Meta

Bases: object

Metadata for a cancel transaction message.

handler_class = 'aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_cancel_handler.TransactionCancelHandler'

message_type = 'transactions/1.0/cancel'

schema_class = 'CancelTransactionSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.cancel_transaction.**CancelTransactionS**

Bases: marshmallow.

Cancel transaction schema class.

class Meta

Bases: object

Cancel transaction schema metadata.

model_class

alias of *aries_cloudagent.protocols.endorse_transaction.v1_0.messages.cancel_transaction.CancelTransaction*

state

thread_id

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.endorsed_transaction_response module

Represents an endorsed transaction message.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.endorsed_transaction_response.**Endors**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing an endorsed transaction response message.

class Meta

Bases: object

Metadata for an endorsed transaction response message.

handler_class = 'aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.endorsed_transaction_response_handler.EndorsedTransactionResponseHandler'

message_type = 'transactions/1.0/endorse'

schema_class = 'EndorsedTransactionResponseSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.endorsed_transaction_response.Endorse

Bases: marshmallow.

Endorsed transaction response schema class.

class Meta

Bases: object

Endorsed transaction response schema metadata.

model_class

alias of *aries_cloudagent.protocols.endorse_transaction.v1_0.messages.endorsed_transaction_response.EndorsedTransactionResponse*

endorser_did

ledger_response

signature_response

state

thread_id

transaction_id

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.messages_attach module

Represents the attached message to be included in the transaction record.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.messages_attach.**MessagesAttach**(*, *au-thor_d* *Op-tional[* = *None*, *au-thor_ve* *Op-tional[* = *None*, *en-dorser_* *Op-tional[* = *None*, *trans-ac-tion_m* *dict* = *{}*, *trans-ac-tion_ty* *Op-tional[* = *None*, *mech-a-nism:* *Op-tional[* = *None*, *taaDi-gest:* *Op-tional[* = *None*, *time:* *Op-tional[* = *None*, ***kwa*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing the attached message.

class Meta
Bases: object

Metadata for attached message class.

message_type = 'transactions/1.0/message'

schema_class = 'MessagesAttachSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.messages_attach.MessagesAttachSchema

Bases: marshmallow.

Attached Message schema class.

class Meta
Bases: object

Attached message schema metadata.

model_class
alias of aries_cloudagent.protocols.endorse_transaction.v1_0.messages.messages_attach.MessagesAttach

data

mime_type

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.refused_transaction_response module

Represents a refused transaction message.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.refused_transaction_response.RefusedT

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a refused transaction response message.

class Meta

Bases: object

Metadata for a refused transaction response message.

handler_class = 'aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.refused_transaction_response_handler.RefusedTransactionResponseHandler'

message_type = 'transactions/1.0/refuse'

schema_class = 'RefusedTransactionResponseSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.refused_transaction_response.RefusedT

Bases: marshmallow.

Refused transaction response schema class.

class Meta

Bases: object

Refused transaction response schema metadata.

model_class
alias of *aries_cloudagent.protocols.endorse_transaction.v1_0.messages.refused_transaction_response.RefusedTransactionResponse*

endorser_did

signature_response

state

thread_id

transaction_id

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_acknowledgement module

Represents a transaction acknowledgement message.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_acknowledgement.**Transacti**

Bases: *aries_cloudagent.protocols.notification.v1_0.messages.ack.V10Ack*

Class representing a transaction acknowledgement message.

class Meta
Bases: object

Metadata for a transaction acknowledgement message.

handler_class = 'aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_acknowledgement_handler.TransactionAcknowledgementHandler'

message_type = 'transactions/1.0/ack'

schema_class = 'TransactionAcknowledgementSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_acknowledgement.**Transacti**

Bases: marshmallow.

Transaction Acknowledgement schema class.

class Meta
Bases: object

Transaction Acknowledgement metadata.

model_class
alias of *aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_acknowledgement.TransactionAcknowledgement*

ledger_response

thread_id

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_job_to_send module

Represents a Transaction Job to send message.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_job_to_send.**TransactionJ**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a transaction job to send.

class Meta
Bases: object

Metadata for a TransactionJobToSend.

handler_class = 'aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_job_to_send_handler.TransactionJobToSendHandler'

message_type = 'transactions/1.0/transaction_my_job'

schema_class = 'TransactionJobToSendSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_job_to_send.**TransactionJ**

Bases: marshmallow.

Transaction Job to send schema class.

class Meta
Bases: object

Metadata for a TransactionJobToSendSchema.

model_class
alias of *aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_job_to_send.TransactionJobToSend*

job

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_request module

Represents a transaction request message.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_request.**TransactionReque**

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a transaction request message.

class Meta

Bases: object

Metadata for a transaction request message.

handler_class = 'aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_request_handler.TransactionRequestHandler'

message_type = 'transactions/1.0/request'

schema_class = 'TransactionRequestSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_request.TransactionReque

Bases: marshmallow.

Transaction request schema class.

class Meta

Bases: object

Transaction request schema metadata.

model_class

alias of *aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_request.TransactionRequest*

endorser_write_txn

messages_attach

signature_request

timing

transaction_id

transaction_type

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_resend module

Represents a transaction resend message.

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_resend.TransactionResend

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a transaction resend message.

class Meta

Bases: object

Metadata for a transaction resend message.

handler_class = 'aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_resend_handler.TransactionResendHandler'

message_type = 'transactions/1.0/resend'

schema_class = 'TransactionResendSchema'

class aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_resend.TransactionResendS

Bases: marshmallow.

Transaction resend schema class.

class Meta

Bases: object

Transaction resend schema metadata.

model_class

alias of *aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_resend.TransactionResend*

state

thread_id

aries_cloudagent.protocols.endorse_transaction.v1_0.models package

Submodules

aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record module

Handle transaction information interface.

class aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.**TransactionRecord**(*,

Represents a single transaction record.

ADD_SIGNATURE = 'add-signature'

CACHE_ENABLED = True

ENDORSE_TRANSACTION = 'aries.transaction.endorse'

FORMAT_VERSION = 'dif/endorse-transaction/request@v1.0'

class Meta

Bases: object

Transaction Record metadata.

schema_class = 'TransactionRecordSchema'

RECORD_ID_NAME = 'transaction_id'

RECORD_TOPIC: Optional[str] = 'endorse_transaction'

RECORD_TYPE = 'transaction'

REFUSE_TRANSACTION = 'aries.transaction.refuse'

REGISTER_PUBLIC_DID = 'aries.transaction.register_public_did'

SIGNATURE_CONTEXT = 'did:sov'

SIGNATURE_REQUEST = 'http://didcomm.org/sign-attachment/%VER/signature-request'

SIGNATURE_RESPONSE = 'http://didcomm.org/sign-attachment/%VER/signature-response'

SIGNATURE_TYPE = '<requested signature type>'

STATE_INIT = 'init'

STATE_REQUEST_RECEIVED = 'request_received'

STATE_REQUEST_SENT = 'request_sent'

STATE_TRANSACTION_ACKED = 'transaction_acked'

STATE_TRANSACTION_CANCELLED = 'transaction_cancelled'

STATE_TRANSACTION_CREATED = 'transaction_created'

STATE_TRANSACTION_ENDORSED = 'transaction_endorsed'

STATE_TRANSACTION_REFUSED = 'transaction_refused'

STATE_TRANSACTION_RESENT = 'transaction_resent'

STATE_TRANSACTION_RESENT_RECEIEVED = 'transaction_resent_received'

TAG_NAMES = {'connection_id', 'state', 'thread_id'}

WRITE_DID_TRANSACTION = 'aries.transaction.ledger.write_did'

WRITE_TRANSACTION = 'aries.transaction.ledger.write'

property record_value: dict

Accessor for the JSON record value generated for this transaction record.

async classmethod retrieve_by_connection_and_thread(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: str*, *thread_id: str*) → *aries_cloudagent.protocols.endorse_transaction.v1_0.models*

Retrieve a TransactionRecord by connection and thread ID.

property transaction_id: str
Accessor for the ID associated with this record.

class aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.TransactionRecordSc

Bases: marshmallow.

Schema to allow serialization/deserialization of transaction records.

class Meta
Bases: object

TransactionRecordSchema metadata.

model_class = 'TransactionRecord'

connection_id

endorser_write_txn

formats

messages_attach

meta_data

signature_request

signature_response

thread_id

timing

transaction_id

Submodules

aries_cloudagent.protocols.endorse_transaction.v1_0.controller module

Protocol controller for endorse transaction.

class aries_cloudagent.protocols.endorse_transaction.v1_0.controller.Controller(*protocol: str*)

Bases: object

Endorse transaction protocol controller.

determine_goal_codes() → Sequence[str]
Return defined goal_codes.

aries_cloudagent.protocols.endorse_transaction.v1_0.manager module

Class to manage transactions.

class aries_cloudagent.protocols.endorse_transaction.v1_0.manager.**TransactionManager**(*profile:* aries_cloudagent.core.pr

Bases: object

Class for managing transactions.

async cancel_transaction(*transaction:* aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.TransactionRec *state: str*)

Cancel a Transaction Request.

Parameters

- **transaction** – The transaction record which would be cancelled
- **state** – The state of the transaction record

Returns The updated transaction and the cancelled transaction response

async complete_transaction(*transaction:* aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.TransactionR *endorser: bool = False*)

Complete a transaction.

This is the final state where the received ledger transaction is written to the ledger.

Parameters **transaction** – The transaction record which would be completed

Returns The updated transaction

async create_endorse_response(*transaction:* aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.Transacti *state: str, use_endorser_did: Optional[str] = None*)

Create a response to endorse a transaction.

Parameters

- **transaction** – The transaction record which would be endorsed.
- **state** – The state of the transaction record

Returns The updated transaction and an endorsed response

async create_record(*messages_attach: str, connection_id: str, meta_data: Optional[dict] = None*)

Create a new Transaction Record.

Parameters

- **messages_attach** – messages to attach, JSON-dumped
- **connection_id** – The connection_id of the ConnRecord between author and endorser

Returns The transaction Record

async create_refuse_response(*transaction:* aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.Transactio *state: str, refuser_did: str*)

Create a response to refuse a transaction.

Parameters

- **transaction** – The transaction record which would be refused

- **state** – The state of the transaction record

Returns The updated transaction and the refused response

async create_request(*transaction: aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.TransactionRecord, signature: Optional[str] = None, signed_request: Optional[dict] = None, expires_time: Optional[str] = None, endorser_write_txn: Optional[bool] = None, author_goal_code: Optional[str] = None, signer_goal_code: Optional[str] = None*)

Create a new Transaction Request.

Parameters

- **transaction** – The transaction from which the request is created.
- **expires_time** – The time till which the endorser should endorse the transaction.

Returns The transaction Record and transaction request

async endorsed_txn_post_processing(*transaction: aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.Tra ledger_response: Optional[dict] = None, connection_record: Optional[aries_cloudagent.connections.models.conn_record.ConnRecord] = None*)

Store record in wallet, and kick off any required post-processing.

Parameters **transaction** – The transaction from which the schema/cred_def would be stored in wallet.

property profile: *aries_cloudagent.core.profile.Profile*

Accessor for the current Profile.

Returns The Profile for this transaction manager

async receive_cancel_transaction(*response: aries_cloudagent.protocols.endorse_transaction.v1_0.messages.cancel_transaction.Ca connection_id: str*)

Update the transaction record to cancel a transaction request.

Parameters

- **response** – The cancel transaction response
- **connection_id** – The connection_id related to this Transaction Record

async receive_endorse_response(*response: aries_cloudagent.protocols.endorse_transaction.v1_0.messages.endorsed_transaction_res*)

Update the transaction record with the endorsed response.

Parameters **response** – The Endorsed Transaction Response

async receive_refuse_response(*response: aries_cloudagent.protocols.endorse_transaction.v1_0.messages.refused_transaction_respo*)

Update the transaction record with a refused response.

Parameters **response** – The refused transaction response

async receive_request(*request: aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_request.TransactionRequ connection_id: str*)

Receive a Transaction request.

Parameters

- **request** – A Transaction Request
- **connection_id** – The connection id related to this transaction record

async **receive_transaction_acknowledgement**(*response:* aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction *connection_id: str*)

Update the transaction record after receiving the transaction acknowledgement.

Parameters

- **response** – The transaction acknowledgement
- **connection_id** – The connection_id related to this Transaction Record

async **receive_transaction_resend**(*response:* aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_resend.Tra *connection_id: str*)

Update the transaction with a resend request.

Parameters

- **response** – The Resend transaction response
- **connection_id** – The connection_id related to this Transaction Record

async **set_transaction_my_job**(*record:* aries_cloudagent.connections.models.conn_record.ConnRecord, *transaction_my_job: str*)

Set transaction_my_job.

Parameters

- **record** – The connection record in which to set transaction jobs
- **transaction_my_job** – My transaction job

Returns The transaction job that is send to other agent

async **set_transaction_their_job**(*tx_job_received:* aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_job_to_sen *receipt:* aries_cloudagent.transport.inbound.receipt.MessageReceipt)

Set transaction_their_job.

Parameters

- **tx_job_received** – The transaction job that is received from the other agent
- **receipt** – The Message Receipt Object

async **transaction_resend**(*transaction:* aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record.TransactionRec *state: str*)

Resend a transaction request.

Parameters

- **transaction** – The transaction record which needs to be resend
- **state** – the state of the transaction record

Returns The updated transaction and the resend response

exception aries_cloudagent.protocols.endorse_transaction.v1_0.manager.**TransactionManagerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Transaction error.

aries_cloudagent.protocols.endorse_transaction.v1_0.message_types module

Message type identifiers for Transactions.

aries_cloudagent.protocols.endorse_transaction.v1_0.routes module

aries_cloudagent.protocols.endorse_transaction.v1_0.transaction_jobs module

Class to manage jobs in Connection Record.

class aries_cloudagent.protocols.endorse_transaction.v1_0.transaction_jobs.**TransactionJob**(*value*)

Bases: enum.Enum

Represents jobs in Connection Record.

TRANSACTION_AUTHOR = (1,)

TRANSACTION_ENDORSER = (2,)

aries_cloudagent.protocols.endorse_transaction.v1_0.util module

Endorser utilities.

async aries_cloudagent.protocols.endorse_transaction.v1_0.util.**get_endorser_connection_id**(*profile: aries_cloudagent.*

Determine default endorser connection for author.

aries_cloudagent.protocols.endorse_transaction.v1_0.util.**is_author_role**(*profile: aries_cloudagent.core.profile.Profile*)

Check if agent is running in author mode.

Submodules

aries_cloudagent.protocols.endorse_transaction.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.introduction package

Subpackages

aries_cloudagent.protocols.introduction.v0_1 package

Subpackages

aries_cloudagent.protocols.introduction.v0_1.handlers package

Submodules

aries_cloudagent.protocols.introduction.v0_1.handlers.forward_invitation_handler module

aries_cloudagent.protocols.introduction.v0_1.handlers.invitation_handler module

Handler for incoming invitation messages.

class aries_cloudagent.protocols.introduction.v0_1.handlers.invitation_handler.**InvitationHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming invitation messages.

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

Message handler implementation.

aries_cloudagent.protocols.introduction.v0_1.handlers.invitation_request_handler module

aries_cloudagent.protocols.introduction.v0_1.messages package

Submodules

aries_cloudagent.protocols.introduction.v0_1.messages.forward_invitation module

Represents a forwarded invitation from another agent.

class aries_cloudagent.protocols.introduction.v0_1.messages.forward_invitation.**ForwardInvitation**(*, *invitation: Optional[a* = *None*, *message: Optional[s* = *None*, ***kwarg*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing an invitation to be forwarded.

class Meta

Bases: object

Metadata for a forwarded invitation.

handler_class = 'aries_cloudagent.protocols.introduction.v0_1.handlers.forward_invitation_handler.ForwardInvitationHandler'

message_type = 'introduction-service/0.1/forward-invitation'

schema_class = 'ForwardInvitationSchema'

class aries_cloudagent.protocols.introduction.v0_1.messages.forward_invitation.**ForwardInvitationSchema**(

Bases: marshmallow.

ForwardInvitation request schema class.

class Meta

Bases: object

ForwardInvitation request schema metadata.

model_class

alias of *aries_cloudagent.protocols.introduction.v0_1.messages.forward_invitation.ForwardInvitation*

invitation

Connection invitation schema class.

aries_cloudagent.protocols.introduction.v0_1.messages.invitation module

Represents an invitation returned to the introduction service.

class aries_cloudagent.protocols.introduction.v0_1.messages.invitation.**Invitation**(*, *invitation: Optional[*aries_cloudagent.pro = None*, *message: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing an invitation returned to the introduction service.

class Meta

Bases: object

Metadata for an invitation.

handler_class = 'aries_cloudagent.protocols.introduction.v0_1.handlers.invitation_handler.InvitationHandler'

message_type = 'introduction-service/0.1/invitation'

schema_class = 'InvitationSchema'

class aries_cloudagent.protocols.introduction.v0_1.messages.invitation.**InvitationSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Invitation request schema class.

class Meta

Bases: object

Invitation request schema metadata.

model_class

alias of *aries_cloudagent.protocols.introduction.v0_1.messages.invitation.Invitation*

invitation

Connection invitation schema class.

aries_cloudagent.protocols.introduction.v0_1.messages.invitation_request module

Represents an request for an invitation from the introduction service.

class aries_cloudagent.protocols.introduction.v0_1.messages.invitation_request.**InvitationRequest**(*, *responder: Optional[st* = *None*, *message: Optional[st* = *None*, ***kwarg*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing an invitation request.

class Meta

Bases: object

Metadata for an invitation request.

handler_class = 'aries_cloudagent.protocols.introduction.v0_1.handlers.invitation_request_handler.InvitationRequestHandler'

message_type = 'introduction-service/0.1/invitation-request'

schema_class = 'InvitationRequestSchema'

class aries_cloudagent.protocols.introduction.v0_1.messages.invitation_request.**InvitationRequestSchema**(

Bases: marshmallow.

Invitation request schema class.

class Meta

Bases: object

Invitation request schema metadata.

model_class

alias of *aries_cloudagent.protocols.introduction.v0_1.messages.invitation_request.InvitationRequest*

Submodules

aries_cloudagent.protocols.introduction.v0_1.base_service module

Introduction service base classes.

class aries_cloudagent.protocols.introduction.v0_1.base_service.**BaseIntroductionService**

Bases: abc.ABC

Service handler for allowing connections to exchange invitations.

abstract async return_invitation(*target_connection_id: str*, *invitation:* aries_cloudagent.protocols.introduction.v0_1.messages.invitation.Invitation, *session:* aries_cloudagent.core.profile.ProfileSession, *outbound_handler*)

Handle the forwarding of an invitation to the responder.

Parameters

- **target_connection_id** – The ID of the connection sending the Invitation
- **invitation** – The received Invitation message
- **session** – Profile session to use for introduction records
- **outbound_handler** – The outbound handler coroutine for sending a message

classmethod service_handler()

Quick accessor for conductor to use.

abstract async start_introduction(*init_connection_id: str*, *target_connection_id: str*, *outbound_handler*, *session:* aries_cloudagent.core.profile.ProfileSession, *message: Optional[str] = None*)

Start the introduction process between two connections.

Parameters

- **init_connection_id** – The connection initiating the request
- **target_connection_id** – The connection which is asked for an invitation
- **outbound_handler** – The outbound handler coroutine for sending a message
- **session** – Profile session to use for connection, introduction records
- **message** – The message to use when requesting the invitation

exception aries_cloudagent.protocols.introduction.v0_1.base_service.**IntroductionError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Generic introduction service error.

aries_cloudagent.protocols.introduction.v0_1.demo_service module

Introduction service demo classes.

class aries_cloudagent.protocols.introduction.v0_1.demo_service.**DemoIntroductionService**
Bases: *aries_cloudagent.protocols.introduction.v0_1.base_service.BaseIntroductionService*

Service handler for allowing connections to exchange invitations.

RECORD_TYPE = 'introduction_record'

async **return_invitation**(*target_connection_id: str*, *invitation:* aries_cloudagent.protocols.introduction.v0_1.messages.invitation.Invitation, *session:* aries_cloudagent.core.profile.ProfileSession, *outbound_handler*)
Handle the forwarding of an invitation to the responder.

Parameters

- **target_connection_id** – The ID of the connection sending the Invitation
- **invitation** – The received (Introduction) Invitation message
- **session** – Profile session to use for introduction records
- **outbound_handler** – The outbound handler coroutine for sending a message

async **start_introduction**(*init_connection_id: str*, *target_connection_id: str*, *message: str*, *session:* aries_cloudagent.core.profile.ProfileSession, *outbound_handler*)
Start the introduction process between two connections.

Parameters

- **init_connection_id** – The connection initiating the request
- **target_connection_id** – The connection which is asked for an invitation
- **outbound_handler** – The outbound handler coroutine for sending a message
- **session** – Profile session to use for connection, introduction records
- **message** – The message to use when requesting the invitation

aries_cloudagent.protocols.introduction.v0_1.message_types module

Message type identifiers for Introductions.

aries_cloudagent.protocols.introduction.v0_1.routes module

Introduction service admin routes.

class aries_cloudagent.protocols.introduction.v0_1.routes.**IntroConnIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Path parameters and validators for request taking connection id.

conn_id

class aries_cloudagent.protocols.introduction.v0_1.routes.**IntroModuleResponseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Response schema for Introduction Module.

class aries_cloudagent.protocols.introduction.v0_1.routes.**IntroStartQueryStringSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Query string parameters for request to start introduction.

message

target_connection_id

aries_cloudagent.protocols.introduction.v0_1.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.introduction.v0_1.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

Submodules

aries_cloudagent.protocols.introduction.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.issue_credential package

Subpackages

aries_cloudagent.protocols.issue_credential.v1_0 package

aries_cloudagent.protocols.issue_credential.v1_0.**problem_report_for_record**(*record: Union[*aries_cloudagent.connections.
aries_cloudagent.protocols.issue_cre
desc_en: str) → *aries_cloudagent.protocols.issue_cre*

Create problem report for record.

Parameters

- **record** – connection or exchange record
- **desc_en** – description text to include in problem report

async aries_cloudagent.protocols.issue_credential.v1_0.**report_problem**(*err:* aries_cloudagent.core.error.BaseError, *desc_en: str*, *http_error_class*, *record:* *Union[*aries_cloudagent.connections.model aries_cloudagent.protocols.issue_credential *outbound_handler:* *Coroutine*)

Send problem report response and raise corresponding HTTP error.

Parameters

- **err** – error for internal diagnostics
- **desc_en** – description text to include in problem report (response)
- **http_error_class** – HTTP error to raise
- **record** – record to cite by thread in problem report
- **outbound_handler** – outbound message handler

Subpackages

aries_cloudagent.protocols.issue_credential.v1_0.handlers package

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_ack_handler module

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_issue_handler module

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_offer_handler module

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_problem_report_handler module

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_proposal_handler module

aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_request_handler module

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.**CredAttrSpec**(*,
n
st
va
st
m
O
ti
=
N
*

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.**CredAttrSpecSc**

Bases: marshmallow.

Attribute preview schema.

class Meta

Bases: object

Attribute preview schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.CredAttrSpec*

mime_type

name

value

class aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.CredentialPrevi

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

Class representing a credential preview inner object.

class Meta
Bases: object

Credential preview metadata.

message_type = 'issue-credential/1.0/credential-preview'

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.CredentialPrevi

Bases: marshmallow.

Credential preview schema.

class Meta
Bases: object

Credential preview schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.inner.credential_preview.CredentialPreview*

attributes

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.**CredentialAck**(***kwargs*)

Bases: *aries_cloudagent.protocols.notification.v1_0.messages.ack.V10Ack*

Class representing a credential ack message.

class Meta

Bases: object

Credential ack metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_ack_handler.CredentialAckHandler'

message_type = 'issue-credential/1.0/ack'

schema_class = 'CredentialAckSchema'

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_ack.**CredentialAckSchema**(**args*, ***kwargs*)

Bases: marshmallow.

Credential ack schema.

class Meta

Bases: object

Schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_ack.CredentialAck*

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_exchange_webhook module

v1.0 credential exchange webhook.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_exchange_webhook.**V10Credenti**

Bases: object

Class representing a state only credential exchange webhook.

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.**CredentialIssue**(*_id: Optional[st = None, *, comment: Optional[st = None, credentials_att Optional[S = None, **kwarg*

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.credential_issue_handler.CredentialIssueHandler'

message_type = 'issue-credential/1.0/issue-credential'

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.**CredentialIssueSchema**(

Bases: marshmallow.

Credential schema.

class Meta

Bases: object

Credential schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_issue.CredentialIssue*

comment

credentials_attach

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.**CredentialOffer**(*_id: Optional[st = None, *, comment: Optional[st = None, credential_prev Optional[a = None, offers_atta Optional[S = None, **kwarg*

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.credential_offer_handler.CredentialOfferHandler'

message_type = 'issue-credential/1.0/offer-credential'

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.AttachDecorator*

Convert an indy credential offer to an attachment decorator.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer.**CredentialOfferSchema**(

Bases: marshmallow.

Credential offer schema.

class Meta

Bases: object

Credential offer schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer.CredentialOffer*

comment

credential_preview

offers_attach

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_problem_report module

A problem report message.

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_problem_report.**CredentialProb**

Bases: *aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReport*

Class representing a problem report message.

class Meta

Bases: object

Problem report metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v1_0.handlers.credential_problem_report_handler.CredentialProblemReportHandler'

message_type = 'issue-credential/1.0/problem-report'

schema_class = 'CredentialProblemReportSchema'

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_problem_report.**CredentialProb**

Bases: marshmallow.

Problem report schema.

class Meta

Bases: object

Schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_problem_report.CredentialProblemReport*

validate_fields(*data*, ***kwargs*)
Validate schema fields.

Parameters **data** – The data to validate

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_problem_report.**ProblemReportR**
Bases: enum.Enum

Supported reason codes.

ISSUANCE_ABANDONED = 'issuance-abandoned'

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.**CredentialProposal**(

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.credential_proposal_handler.CredentialProposalHandler'

message_type = 'issue-credential/1.0/propose-credential'

schema_class = 'CredentialProposalSchema'

class aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_proposal.CredentialProposalS

Bases: marshmallow.

Credential proposal schema.

class Meta
Bases: object

Credential proposal schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_proposal.CredentialProposal*

comment

cred_def_id

credential_proposal

issuer_did

schema_id

schema_issuer_did

schema_name

schema_version

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.**CredentialRequest**(*_id Op tio = No *, co me Op tio = No re- que Op tio = No **p

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.credential_request_handler.CredentialRequestHandler'

message_type = 'issue-credential/1.0/request-credential'

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.**CredentialRequestSch**

Bases: marshmallow.

Credential request schema.

class Meta

Bases: object

Credential request schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_request.CredentialRequest*

comment

requests_attach

aries_cloudagent.protocols.issue_credential.v1_0.models package

Package-wide code and data.

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.V10CredentialExchange

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_TOPIC: Optional[str] = 'issue_credential'

RECORD_TYPE = 'credential_exchange_v10'

ROLE_HOLDER = 'holder'

ROLE_ISSUER = 'issuer'

STATE_ABANDONED = 'abandoned'

STATE_ACKED = 'credential_acked'

STATE_CREDENTIAL_RECEIVED = 'credential_received'

STATE_CREDENTIAL_REVOKED = 'credential_revoked'

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'}

property credential: *aries_cloudagent.indy.models.cred_precis.IndyCredInfo*
Accessor; get deserialized view.

property credential_exchange_id: str
Accessor for the ID associated with this exchange.

property credential_offer:
aries_cloudagent.indy.models.cred_abstract.IndyCredAbstract
Accessor; get deserialized view.

property credential_offer_dict: *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer.CredentialOffer*
Accessor; get deserialized view.

property credential_proposal_dict: *aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_proposal.CredentialProposal*
Accessor; get deserialized view.

property credential_request:
aries_cloudagent.indy.models.cred_request.IndyCredRequest
Accessor; get deserialized view.

async emit_event(*session:* aries_cloudagent.core.profile.ProfileSession, *payload: Optional[Any] = None*)
Emit an event.

Parameters

- **session** – The profile session to use
- **payload** – The event payload

property raw_credential: *aries_cloudagent.indy.models.cred.IndyCredential*
Accessor; get deserialized view.

property record_value: dict
Accessor for the JSON record value generated for this invitation.

async classmethod retrieve_by_connection_and_thread(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: Optional[str]*, *thread_id: str*, *role: Optional[str] = None*, ***, *for_update=False*) → *aries_cloudagent.protocols.issue_credential.v1_0.models.cre*
Retrieve a credential exchange record by connection and thread ID.

async save_error_state(*session:* aries_cloudagent.core.profile.ProfileSession, ***, *state: Optional[str] = None*, *reason: Optional[str] = None*, *log_params: Optional[Mapping[str, Any]] = None*, *log_override: bool = False*)
Save record error state if need be; log and swallow any storage error.

Parameters

- **session** – The profile session to use
- **reason** – A reason to add to the log
- **log_params** – Additional parameters to log
- **override** – Override configured logging regimen, print to stderr instead

class aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.**V10CredentialExchange**

Bases: marshmallow.

Schema to allow serialization/deserialization of credential exchange records.

class Meta
Bases: object

V10CredentialExchangeSchema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange.V10CredentialExchange*

auto_issue

auto_offer

auto_remove

connection_id

credential

credential_definition_id

credential_exchange_id

credential_id

credential_offer

credential_offer_dict

credential_proposal_dict

credential_request

credential_request_metadata

error_msg

initiator

parent_thread_id

raw_credential

revoc_reg_id

revocation_id

role

schema_id

state

thread_id

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.controller module

Protocol controller for issue credential v1_0.

class aries_cloudagent.protocols.issue_credential.v1_0.controller.**Controller**(*protocol: str*)
Bases: object

Issue credential v1_0 protocol controller.

determine_goal_codes() → Sequence[str]
Return defined goal_codes.

aries_cloudagent.protocols.issue_credential.v1_0.manager module

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

aries_cloudagent.protocols.issue_credential.v2_0 package

aries_cloudagent.protocols.issue_credential.v2_0.**problem_report_for_record**(*record: Union[*aries_cloudagent.connections.
aries_cloudagent.protocols.issue_cre
desc_en: str) → *aries_cloudagent.protocols.issue_cre*

Create problem report for record.

Parameters

- **record** – connection or exchange record
- **desc_en** – description text to include in problem report

async aries_cloudagent.protocols.issue_credential.v2_0.**report_problem**(*err:* aries_cloudagent.core.error.BaseError, *desc_en: str*, *http_error_class*, *record: Union[*aries_cloudagent.connections.model
aries_cloudagent.protocols.issue_credential
outbound_handler: Coroutine)

Send problem report response and raise corresponding HTTP error.

Parameters

- **err** – error for internal diagnostics
- **desc_en** – description text to include in problem report (response)
- **http_error_class** – HTTP error to raise
- **record** – record to cite by thread in problem report
- **outbound_handler** – outbound message handler

Subpackages

aries_cloudagent.protocols.issue_credential.v2_0.formats package

Subpackages

aries_cloudagent.protocols.issue_credential.v2_0.formats.indy package

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.formats.indy.handler module

aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof package

Subpackages

aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models package

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail module

Linked data proof verifiable options detail artifacts to attach to RFC 453 messages.

class aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail.**LDProofVCDeta**

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

Linked data proof verifiable credential detail.

class Meta

Bases: object

LDProofVCDetail metadata.

schema_class = 'LDProofVCDetailSchema'

class aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail.**LDProofVCDeta**

Bases: marshmallow.

Linked data proof verifiable credential detail schema.

class Meta

Bases: object

Accept parameter overload.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail.LDProofVCDetail*

credential

options

aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail_options module

LDProofVCDetailOptions.

class aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail_options.**Cred**

Bases: marshmallow.

Linked data proof credential status options schema.

class Meta

Bases: object

Accept parameter overload.

type

class aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail_options.**LDPr**

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

Linked Data Proof verifiable credential options model.

class Meta

Bases: object

LDProofVCDetailOptions metadata.

schema_class = 'LDProofVCDetailOptionsSchema'

class aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail_options.**LDPr**

Bases: marshmallow.

Linked data proof verifiable credential options schema.

class Meta

Bases: object

Accept parameter overload.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail_options.LDProofVCDetailOptions*

challenge

created

credential_status

domain

proof_purpose

proof_type

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.handler module

V2.0 issue-credential linked data proof credential format handler.

class aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.handler.**LDProofCredFormatHandle**

Bases: *aries_cloudagent.protocols.issue_credential.v2_0.formats.handler.V20CredFormatHandler*

Linked data proof credential format handler.

async create_offer(*cred_proposal_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create linked data proof credential offer.

async create_proposal(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *proposal_data: Mapping*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create linked data proof credential proposal.

async create_request(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *request_data: Optional[Mapping] = None*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create linked data proof credential request.

format: *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format* **= FormatSpec(aries='aries/', detail=<class 'aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof.V20CredExRecordLDProof'>, handler=<aries_cloudagent.utils.classloader.DeferLoad object>)**

async get_detail_record(*cred_ex_id: str*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof.V20CredExRecordLDPr*

Retrieve credential exchange detail record by cred_ex_id.

get_format_data(*message_type: str*, *data: dict*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Get credential format and attachment objects for use in cred ex messages.

Returns a tuple of both credential format and attachment decorator for use in credential exchange messages. It looks up the correct format identifier and encodes the data as a base64 attachment.

Parameters

- **message_type** (*str*) – The message type for which to return the cred format. Should be one of the message types defined in the message types file
- **data** (*dict*) – The data to include in the attach decorator

Returns Credential format and attachment data objects

Return type CredFormatAttachment

get_format_identifier(*message_type: str*) → str

Get attachment format identifier for format and message combination.

Parameters **message_type** (*str*) – Message type for which to return the format identifier

Returns Issue credential attachment format identifier

Return type str

async issue_credential(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *retries: int = 5*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Issue linked data proof credential.

async receive_credential(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *cred_issue_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.V20CredIssue) → None

Receive linked data proof credential.

async receive_offer(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *cred_offer_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.V20CredOffer) → None

Receive linked data proof credential offer.

async receive_proposal(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *cred_proposal_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal) → None

Receive linked data proof credential proposal.

async receive_request(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *cred_request_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.V20CredRequest) → None

Receive linked data proof request.

async store_credential(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *cred_id: Optional[str] = None*) → None

Store linked data proof credential.

classmethod validate_fields(*message_type: str*, *attachment_data: Mapping*) → None

Validate attachment data for a specific message type.

Uses marshmallow schemas to validate if format specific attachment data is valid for the specified message type. Only does structural and type checks, does not validate if .e.g. the issuer value is valid.

Parameters

- **message_type** (*str*) – The message type to validate the attachment data for. Should be one of the message types as defined in message_types.py
- **attachment_data** (*Mapping*) – [description] The attachment data to valide

Raises **Exception** – When the data is not valid.

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.formats.handler module

V2.0 issue-credential base credential format handler.

exception aries_cloudagent.protocols.issue_credential.v2_0.formats.handler.**V20CredFormatError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Credential format error under issue-credential protocol v2.0.

class aries_cloudagent.protocols.issue_credential.v2_0.formats.handler.**V20CredFormatHandler**(*profile:* aries_cloudage

Bases: abc.ABC

Base credential format handler.

abstract async create_offer(*cred_proposal_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredForma*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create format specific credential offer attachment data.

abstract async create_proposal(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRe*proposal_data: Mapping*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFo*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create format specific credential proposal attachment data.

abstract async create_request(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRec*request_data: Optional[Mapping] = None*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFor*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create format specific credential request attachment data.

format: *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format* = **None**

abstract get_format_data(*message_type: str*, *data: dict*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Get credential format and attachment objects for use in cred ex messages.

abstract get_format_identifier(*message_type: str*) → str

Get attachment format identifier for format and message combination.

Parameters **message_type** (str) – Message type for which to return the format identifier

Returns Issue credential attachment format identifier

Return type str

abstract async issue_credential(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExR *retries: int = 5*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredF* *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create format specific issue credential attachment data.

property profile: *aries_cloudagent.core.profile.Profile*

Accessor for the current profile instance.

Returns The profile instance for this credential format

abstract async receive_credential(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredE *cred_issue_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.V20CredIssue → None

Create format specific issue credential message.

abstract async receive_offer(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExReco *cred_offer_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.V20CredOffer) → None

Receive foramt specific credential offer message.

abstract async receive_proposal(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExR *cred_proposal_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredPro → None

Receive format specific credential proposal message.

abstract async receive_request(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRe *cred_request_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.V20CredReque → None

Receive format specific credential request message.

abstract async store_credential(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExR *cred_id: Optional[str] = None*) → None

Store format specific credential from issue credential message.

abstract classmethod validate_fields(*message_type: str*, *attachment_data: dict*) → None

Validate attachment data for specific message type and format.

aries_cloudagent.protocols.issue_credential.v2_0.handlers package

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_ack_handler module

Credential ack message handler.

class aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_ack_handler.**V20CredAckHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential acks.

async 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.v2_0.handlers.cred_issue_handler module

Credential issue message handler.

class aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_issue_handler.**V20CredIssueHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential offers.

async 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.v2_0.handlers.cred_offer_handler module

Credential offer message handler.

class aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_offer_handler.**V20CredOfferHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential offers.

async 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.v2_0.handlers.cred_problem_report_handler module

Credential problem report message handler.

class aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_problem_report_handler.**CredProblemReportHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for problem reports.

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

Message handler logic for problem reports.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_proposal_handler module

Credential proposal message handler.

class aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_proposal_handler.**V20CredProposalHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential proposals.

async 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.v2_0.handlers.cred_request_handler module

Credential request message handler.

class aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_request_handler.**V20CredRequestHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for credential requests.

async 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.v2_0.messages package

Subpackages

aries_cloudagent.protocols.issue_credential.v2_0.messages.inner package

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview module

Credential preview inner object.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview.**V20CredAttrSpec**(*, *name*: *str*, *value*: *str*, *mime_type*: *Optional* = *None*, ***kwa*

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

Attribute preview.

class Meta

Bases: object

Attribute preview metadata.

schema_class = 'V20CredAttrSpecSchema'

b64_decoded_value() → str

Value, base64-decoded if applicable.

static list_plain(*plain: dict*) → Sequence[*aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview.V20CredAttrSpec*]

Return a list of *V20CredAttrSpec* (copies), absent any MIME types.

Parameters **plain** – dict mapping names to values

Returns List of *V20CredAttrSpec* (copies), absent any MIME types

class aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview.**V20CredAttrSpecSchema**

Bases: marshmallow.

Attribute preview schema.

class Meta
Bases: object

Attribute preview schema metadata.

model_class
alias of aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview.V20CredAttrSpec

mime_type

name

value

class aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview.V20CredPreview(*, _type: Optional[= None, attribute Optional[= None, **kwa

Bases: aries_cloudagent.messaging.models.base.BaseModel

Credential preview.

class Meta
Bases: object

Credential preview metadata.

message_type = 'issue-credential/2.0/credential-preview'

schema_class = 'V20CredPreviewSchema'

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.v2_0.messages.inner.cred_preview.V20CredPreviewSchema

Bases: marshmallow.

Credential preview schema.

class Meta
Bases: object

Credential preview schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview.V20CredPreview*

attributes

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ack module

Credential ack message.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ack.**V20CredAck**(***kwargs*)
Bases: *aries_cloudagent.protocols.notification.v1_0.messages.ack.V10Ack*

Credential ack.

class Meta
Bases: object

Credential ack metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_ack_handler.V20CredAckHandler'

message_type = 'issue-credential/2.0/ack'

schema_class = 'V20CredAckSchema'

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ack.**V20CredAckSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Credential ack schema.

class Meta
Bases: object

Schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ack.V20CredAck*

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ex_record_webhook module

v2.0 credential exchange webhook.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ex_record_webhook.**V20CredExRecordW**
Bases: object

Class representing a state only credential exchange webhook.

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format module

Issue-credential protocol message attachment format.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.**FormatSpec**(*aries*, *detail*, *handler*)

Bases: tuple

property aries
Alias for field number 0

property detail
Alias for field number 1

property handler
Alias for field number 2

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.**V20CredFormat**(*, *attach_id: Optional[str] = None*, *format_: Optional[str] = None*)

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

Issue-credential protocol message attachment format.

class Format(*value*)
Bases: enum.Enum

Attachment format.

INDY = FormatSpec(aries='hlindy/', detail=<class 'aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy.V20CredExRecordIndy'>, handler=<aries_cloudagent.utils.classloader.DeferLoad object>)

LD_PROOF = FormatSpec(aries='aries/', detail=<class 'aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof.V20CredExRecordLDProof'>, handler=<aries_cloudagent.utils.classloader.DeferLoad object>)

property api: str
Admin API specifier.

property aries: str
Aries specifier prefix.

property detail: Union[*aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy.V20CredExRecordIndy*, *aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof.V20CredExRecordLDProof*]

Accessor for credential exchange detail class.

classmethod get(*label: Union[str, aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format]*)
Get format enum for label.

get_attachment_data(*formats: Sequence[aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*, *attachments: Sequence[aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator]*)
Find attachment of current format, decode and return its content.

property handler: Type[V20CredFormatHandler]
Accessor for credential exchange format handler.

validate_fields(*message_type: str*, *attachment_data: Mapping*)
Raise ValidationError for invalid attachment formats.

class Meta
Bases: object

Issue-credential protocol message attachment format metadata.

schema_class = 'V20CredFormatSchema'

property format: str
Return format.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.**V20CredFormatSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Issue-credential protocol message attachment format schema.

class Meta
Bases: object

Issue-credential protocol message attachment format schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat*

attach_id

format_

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue module

Credential issue message.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.**V20CredIssue**(*_id: Optional[str] = None*, ***, *replacement_id: Optional[str] = None*, *comment: Optional[str] = None*, *formats: Optional[Sequence[ari = None*, *credentials_attach: Optional[Sequence[ari = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Credential issue message.

class Meta

Bases: object

V20CredIssue metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_issue_handler.V20CredIssueHandler'

message_type = 'issue-credential/2.0/issue-credential'

schema_class = 'V20CredIssueSchema'

attachment(*fmt: Optional[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format*] = None*) → dict

Return attached credential.

Parameters **fmt** – format of attachment in list to decode and return

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.**V20CredIssueSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Credential issue schema.

class Meta

Bases: object

Credential issue schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.V20CredIssue*

comment

credentials_attach

formats

replacement_id

validate_fields(*data*, ***kwargs*)

Validate attachments per format.

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer module

Credential offer message.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.**V20CredOffer**(*_id: Optional[str] = None*, *, *replacement_id: Optional[str] = None*, *comment: Optional[str] = None*, *credential_preview: Optional[aries_cloudag = None*, *formats: Optional[Sequence[ari = None*, *offers_attach: Optional[Sequence[ari = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Credential offer.

class **Meta**

Bases: object

V20CredOffer metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_offer_handler.V20CredOfferHandler'

message_type = 'issue-credential/2.0/offer-credential'

schema_class = 'V20CredOfferSchema'

attachment(*fmt: Optional[aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format] = None*) → dict

Return attached offer.

Parameters **fmt** – format of attachment in list to decode and return

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.**V20CredOfferSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Credential offer schema.

class Meta

Bases: object

Credential offer schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.V20CredOffer*

comment

credential_preview

formats

offers_attach

replacement_id

validate_fields(*data*, ***kwargs*)

Validate attachments per format.

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_problem_report module

A problem report message.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_problem_report.**ProblemReportReason**

Bases: enum.Enum

Supported reason codes.

ISSUANCE_ABANDONED = 'issuance-abandoned'

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_problem_report.**V20CredProblemRepor**

Bases: *aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReport*

Class representing a problem report message.

class Meta

Bases: object

Problem report metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_problem_report_handler.CredProblemReportHandler'

message_type = 'issue-credential/2.0/problem-report'

schema_class = 'V20CredProblemReportSchema'

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_problem_report.V20CredProblemRepor

Bases: marshmallow.

Problem report schema.

class Meta
Bases: object

Schema metadata.

model_class
alias of aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_problem_report.V20CredProblemReport

validate_fields(*data*, ***kwargs*)
Validate schema fields.

Parameters **data** – The data to validate

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal module

Credential proposal message.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.**V20CredProposal**(*_id: Optional[str] = None*, *, *comment: Optional[str] = None*, *credential_preview Optional[aries_ = None*, *formats: Optional[Seque = None*, *filters_attach: Optional[Seque = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Credential proposal.

class Meta

Bases: object

V20CredProposal metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_proposal_handler.V20CredProposalHandler'

message_type = 'issue-credential/2.0/propose-credential'

schema_class = 'V20CredProposalSchema'

attachment(*fmt: Optional[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format*] = None*) → dict

Return attached filter.

Parameters **fmt** – format of attachment in list to decode and return

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.**V20CredProposalSchema**(*arg
Any,
**kw
Any)

Bases: marshmallow.

Credential proposal schema.

class Meta

Bases: object

Credential proposal schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal*

comment

credential_preview

filters_attach

formats

validate_fields(*data*, ***kwargs*)

Validate attachments per format.

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request module

Credential request message.

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.**V20CredRequest**(*_id:
Optional[str]
=
None,
*,
*comment:
Optional[str]
=
None,
*formats:
Optional[Sequenc
=
None,
*requests_attach:
Optional[Sequenc
=
None,
***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Credential request.

class Meta

Bases: object

V20CredRequest metadata.

handler_class = 'aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_request_handler.V20CredRequestHandler'

message_type = 'issue-credential/2.0/request-credential'

schema_class = 'V20CredRequestSchema'

attachment(*fmt: Optional[aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format] = None*) → dict

Return attached credential request.

Parameters **fmt** – format of attachment in list to decode and return

class aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.**V20CredRequestSchema**(**args: Any, **kwa Any*)

Bases: marshmallow.

Credential request schema.

class Meta

Bases: object

Credential request schema metadata.

model_class

alias of *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.V20CredRequest*

comment

formats

requests_attach

validate_fields(*data*, ***kwargs*)

Validate attachments per format.

aries_cloudagent.protocols.issue_credential.v2_0.models package

Package-wide code and data.

Subpackages

aries_cloudagent.protocols.issue_credential.v2_0.models.detail package

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy module

Indy-specific credential exchange information with non-secrets storage.

class aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy.**V20CredExRecordIndy**(*cred_ex_ind* *Optional[str]* = *None*, *, *cred_ex_id: Optional[str]* = *None*, *cred_id_sto Optional[str]* = *None*, *cred_reques Optional[Mapp* = *None*, *rev_reg_id: Optional[str]* = *None*, *cred_rev_id Optional[str]* = *None*, ***kwargs*)

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

Credential exchange indy detail record.

class Meta

Bases: object

V20CredExRecordIndy metadata.

schema_class = 'V20CredExRecordIndySchema'

RECORD_ID_NAME = 'cred_ex_indy_id'

RECORD_TOPIC: Optional[str] = 'issue_credential_v2_0_indy'

RECORD_TYPE = 'indy_cred_ex_v20'

TAG_NAMES = {'cred_ex_id'}

property cred_ex_indy_id: str
Accessor for the ID associated with this exchange.

async classmethod query_by_cred_ex_id(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_ex_id: str*) → Sequence[*aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy.V2*
Retrieve credential exchange indy detail record(s) by its cred ex id.

property record_value: dict
Accessor for the JSON record value generated for this credential exchange.

class aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy.**V20CredExRecordIndySchema**(**arg*
Any,
***k*
Any)

Bases: marshmallow.

Credential exchange indy detail record detail schema.

class Meta
Bases: object

Credential exchange indy detail record schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy.V20CredExRecordIndy*

cred_ex_id

cred_ex_indy_id

cred_id_stored

cred_request_metadata

cred_rev_id

rev_reg_id

aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof module

Linked data proof specific credential exchange information with non-secrets storage.

class aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof.**V20CredExRecordLDProof**(*cre*
Op
tio
=
No
**,*
cre
Op
tio
=
No
cre
Op
tio
=
No
***j*

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

Credential exchange linked data proof detail record.

class Meta

Bases: object

V20CredExRecordLDProof metadata.

schema_class = 'V20CredExRecordLDProofSchema'

RECORD_ID_NAME = 'cred_ex_ld_proof_id'

RECORD_TOPIC: Optional[str] = 'issue_credential_v2_0_ld_proof'

RECORD_TYPE = 'ld_proof_cred_ex_v20'

TAG_NAMES = {'cred_ex_id'}

property cred_ex_ld_proof_id: str

Accessor for the ID associated with this exchange.

async classmethod query_by_cred_ex_id(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_ex_id: str*) → Sequence[*aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proc*

Retrieve a credential exchange LDProof detail record by its cred ex id.

property record_value: dict

Accessor for the JSON record value generated for this credential exchange.

class aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof.**V20CredExRecordLDProofSch**

Bases: marshmallow.

Credential exchange linked data proof detail record detail schema.

class Meta

Bases: object

Credential exchange linked data proof detail record schema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof.V20CredExRecordLDProof*

cred_ex_id

cred_ex_ld_proof_id

cred_id_stored

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record module

Aries#0453 v2.0 credential exchange information with non-secrets storage.

class aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.**V20CredExRecord**(*, *cred_ex_id: Optional[str] = None*, *connection_id: Optional[str] = None*, *verification_method: Optional[str] = None*, *thread_id: Optional[str] = None*, *parent_thread_id Optional[str] = None*, *initiator: Optional[str] = None*, *role: Optional[str] = None*, *state: Optional[str] = None*, *cred_proposa Optional[Union[aries_cloudag = None*, *cred_offer: Optional[Union[aries_cloudag*

Represents an Aries#0036 credential exchange.

INITIATOR_EXTERNAL = 'external'

INITIATOR_SELF = 'self'

class Meta
Bases: object

CredentialExchange metadata.

schema_class = 'V20CredExRecordSchema'

RECORD_ID_NAME = 'cred_ex_id'

RECORD_TOPIC: Optional[str] = 'issue_credential_v2_0'

RECORD_TYPE = 'cred_ex_v20'

ROLE_HOLDER = 'holder'

ROLE_ISSUER = 'issuer'

STATE_ABANDONED = 'abandoned'

STATE_CREDENTIAL_RECEIVED = 'credential-received'

STATE_CREDENTIAL_REVOKED = 'credential-revoked'

STATE_DONE = 'done'

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'}

property by_format: Mapping
Record proposal, offer, request, and credential attachments by format.

property cred_ex_id: str
Accessor for the ID associated with this exchange.

property cred_issue:
aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.V20CredIssue
Accessor; get deserialized view.

property cred_offer:
aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.V20CredOffer
Accessor; get deserialized view.

property cred_preview: Mapping
Credential preview (deserialized view) from credential proposal.

property cred_proposal: *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal*
Accessor; get deserialized view.

property cred_request: *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.V20CredRequest*
Accessor; get deserialized view.

async emit_event(*session:* aries_cloudagent.core.profile.ProfileSession, *payload: Optional[Any] = None*)
Emit an event.

Parameters

- **session** – The profile session to use
- **payload** – The event payload

property record_value: Mapping
Accessor for the JSON record value generated for this credential exchange.

async classmethod retrieve_by_conn_and_thread(*session:* aries_cloudagent.core.profile.ProfileSession, *connection_id: Optional[str]*, *thread_id: str*, *role: Optional[str] = None*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_re*
Retrieve a credential exchange record by connection and thread ID.

async save_error_state(*session:* aries_cloudagent.core.profile.ProfileSession, ***, *state: Optional[str] = None*, *reason: Optional[str] = None*, *log_params: Optional[Mapping[str, Any]] = None*, *log_override: bool = False*)
Save record error state if need be; log and swallow any storage error.

Parameters

- **session** – The profile session to use
- **reason** – A reason to add to the log
- **log_params** – Additional parameters to log
- **override** – Override configured logging regimen, print to stderr instead

class aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.**V20CredExRecordSchema**(**args Any*, ***kw Any*)

Bases: marshmallow.

Schema to allow serialization/deserialization of credential exchange records.

class Meta
Bases: object

V20CredExSchema metadata.

model_class
alias of *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*

auto_issue

auto_offer

auto_remove

by_format

connection_id

cred_ex_id

cred_issue

cred_offer

cred_preview

cred_proposal

cred_request

error_msg

initiator

parent_thread_id

role

state

thread_id

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.controller module

Protocol controller for issue credential v2_0.

class aries_cloudagent.protocols.issue_credential.v2_0.controller.**Controller**(*protocol: str*)

Bases: object

Issue credential v2_0 protocol controller.

determine_goal_codes() → Sequence[str]

Return defined goal_codes.

aries_cloudagent.protocols.issue_credential.v2_0.manager module

V2.0 issue-credential protocol manager.

class aries_cloudagent.protocols.issue_credential.v2_0.manager.**V20CredManager**(*profile:* aries_cloudagent.core.profile.Pro

Bases: object

Class for managing credentials.

async create_offer(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *counter_proposal: Optional[*aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal*] = None*, *replacement_id: Optional[str] = None*, *comment: Optional[str] = None*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*, *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.V20CredOffer*]

Create credential offer, update credential exchange record.

Parameters

- **cred_ex_record** – credential exchange record for which to create offer

- **replacement_id** – identifier to help coordinate credential replacement
- **comment** – optional human-readable comment to set in offer message

Returns A tuple (credential exchange record, credential offer message)

async create_proposal(*connection_id: str*, *, *auto_remove: Optional[bool] = None*, *comment: Optional[str] = None*, *cred_preview: aries_cloudagent.protocols.issue_credential.v2_0.messages.inner.cred_preview.V20CredPreview*, *fmt2filter: Mapping[aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format.V20CredFormat.Format, Mapping[str, str]]*, *trace: bool = False*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*

Create a credential proposal.

Parameters

- **connection_id** – connection for which to create proposal
- **auto_remove** – whether to remove record automatically on completion
- **comment** – optional human-readable comment to include in proposal
- **cred_preview** – credential preview to use to create credential proposal
- **fmt2filter** – mapping between format and filter
- **trace** – whether to trace the operation

Returns Resulting credential exchange record including credential proposal

async create_request(*cred_ex_record: aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*, *holder_did: str*, *comment: Optional[str] = None*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*, *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.V20CredRequest*]

Create a credential request.

Parameters

- **cred_ex_record** – credential exchange record for which to create request
- **holder_did** – holder DID
- **comment** – optional human-readable comment to set in request message

Returns A tuple (credential exchange record, credential request message)

async delete_cred_ex_record(*cred_ex_id: str*) → None

Delete credential exchange record and associated detail records.

async issue_credential(*cred_ex_record: aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*, *, *comment: Optional[str] = None*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*, *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.V20CredIssue*]

Issue a credential.

Parameters

- **cred_ex_record** – credential exchange record for which to issue credential
- **comment** – optional human-readable comment pertaining to credential issue

Returns (Updated credential exchange record, credential issue message)

Return type Tuple

async prepare_send(*connection_id: str*, *cred_proposal:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal, *verification_method: Optional[str] = None*, *auto_remove: Optional[bool] = None*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*, *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.V20CredOffer*]

Set up a new credential exchange record for an automated send.

Parameters

- **connection_id** – connection for which to create offer
- **cred_proposal** – credential proposal with preview
- **verification_method** – an optional verification method to be used when issuing
- **auto_remove** – flag to remove the record automatically on completion

Returns A tuple of the new credential exchange record and credential offer message

property profile: *aries_cloudagent.core.profile.Profile*

Accessor for the current profile instance.

Returns The profile instance for this credential manager

async receive_credential(*cred_issue_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue.V20CredIssue, *connection_id: Optional[str]*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*

Receive a credential issue message from an issuer.

Hold cred in storage potentially to be processed by controller before storing.

Returns Credential exchange record, retrieved and updated

async receive_credential_ack(*cred_ack_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ack.V20CredAck, *connection_id: Optional[str]*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExReco*

Receive credential ack from holder.

Parameters

- **cred_ack_message** – credential ack message to receive
- **connection_id** – connection identifier

Returns credential exchange record, retrieved and updated

async receive_offer(*cred_offer_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer.V20CredOffer, *connection_id: Optional[str]*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*

Receive a credential offer.

Parameters

- **cred_offer_message** – credential offer message
- **connection_id** – connection identifier

Returns The credential exchange record, updated

async receive_problem_report(*message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_problem_report.V20Cred*connection_id: str*)

Receive problem report.

Returns credential exchange record, retrieved and updated

async receive_proposal(*cred_proposal_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal.V20CredProposal, *connection_id: str*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*

Receive a credential proposal.

Returns The resulting credential exchange record, created

async receive_request(*cred_request_message:* aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request.V20CredRequest, *connection_record: Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]*, *oob_record: Optional[*aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record.OobRecord*]*) → *aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*

Receive a credential request.

Parameters

- **cred_request_message** – credential request to receive
- **connection_id** – connection identifier

Returns credential exchange record, updated

async send_cred_ack(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord)

Create, send, and return ack message for input cred ex record.

Delete cred ex record if set to auto-remove.

Returns cred ex record, cred ack message for tracing

Return type Tuple

async store_credential(*cred_ex_record:* aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord, *cred_id: Optional[str] = None*) → Tuple[*aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record.V20CredExRecord*, *aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ack.V20CredAck*]

Store a credential in holder wallet; send ack to issuer.

Parameters

- **cred_ex_record** – credential exchange record with credential to store and ack
- **cred_id** – optional credential identifier to override default on storage

Returns Updated credential exchange record

exception aries_cloudagent.protocols.issue_credential.v2_0.manager.**V20CredManagerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Credential manager error under issue-credential protocol v2.0.

aries_cloudagent.protocols.issue_credential.v2_0.message_types module

Message and inner object type identifiers for Connections.

aries_cloudagent.protocols.issue_credential.v2_0.routes module

Credential exchange admin routes.

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredBoundOfferRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending bound credential offer admin message.

counter_preview

filter_

validate_fields(*data*, ***kwargs*)

Validate schema fields: need both filter and counter_preview or neither.

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredExFreeSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending credential admin message.

connection_id

verification_method

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredExIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters and validators for request taking credential exchange id.

cred_ex_id

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredExRecordDetailSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Credential exchange record and any per-format details.

cred_ex_record

indy

ld_proof

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredExRecordListQueryStringSchema**(**args Any*, ***kw Any*)

Bases: marshmallow.

Parameters and validators for credential exchange record list query.

connection_id

role

state

thread_id

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredExRecordListResultSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for credential exchange record list query.

results

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredFilterIndySchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Indy credential filtration criteria.

cred_def_id

issuer_did

schema_id

schema_issuer_did

schema_name

schema_version

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredFilterLDProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Credential filtration criteria.

ld_proof

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredFilterSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Credential filtration criteria.

indy

ld_proof

validate_fields(*data*, ***kwargs*)

Validate schema fields.

Data must have indy, ld_proof, or both.

Parameters **data** – The data to validate

Raises **ValidationError** – if data has neither indy nor ld_proof

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters and validators for request taking credential id.

credential_id

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredIssueProblemReportRequestSchema**(**a*
An
***k*
An

Bases: marshmallow.

Request schema for sending problem report.

description

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredIssueRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending credential issue admin message.

comment

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredOfferConnFreeRequestSchema**(**args: Any*, ***kwargs*
Any)

Bases: marshmallow.

Request schema for creating credential offer free from connection.

auto_issue

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredOfferRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending credential offer admin message.

auto_issue

connection_id

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredRequestFreeSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Filter, auto-remove, comment, trace.

auto_remove

comment

connection_id

filter_

holder_did

trace

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredRequestRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending credential request message.

holder_did

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20CredStoreRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending a credential store admin message.

credential_id

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20IssueCredSchemaCore**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Filter, auto-remove, comment, trace.

auto_remove

comment

credential_preview

filter_

validate(*data*, ***kwargs*)
Make sure preview is present when indy format is present.

class aries_cloudagent.protocols.issue_credential.v2_0.routes.**V20IssueCredentialModuleResponseSchema**(**a*
An
***k*
An

Bases: marshmallow.

Response schema for v2.0 Issue Credential Module.

aries_cloudagent.protocols.issue_credential.v2_0.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.issue_credential.v2_0.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

Submodules

aries_cloudagent.protocols.issue_credential.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.notification package

Subpackages

aries_cloudagent.protocols.notification.v1_0 package

Subpackages

aries_cloudagent.protocols.notification.v1_0.handlers package

Submodules

aries_cloudagent.protocols.notification.v1_0.handlers.ack_handler module

Generic ack message handler.

class aries_cloudagent.protocols.notification.v1_0.handlers.ack_handler.**V10AckHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for generic acks.

async 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.notification.v1_0.messages package

Submodules

aries_cloudagent.protocols.notification.v1_0.messages.ack module

Represents an explicit RFC 15 ack message, adopted into present-proof protocol.

class aries_cloudagent.protocols.notification.v1_0.messages.ack.**V10Ack**(*status: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Base class representing an explicit ack message for no specific protocol.

class Meta

Bases: object

V10Ack metadata.

handler_class = 'aries_cloudagent.protocols.notification.v1_0.handlers.ack_handler.V10AckHandler'

message_type = 'notification/1.0/ack'

schema_class = 'V10AckSchema'

class aries_cloudagent.protocols.notification.v1_0.messages.ack.**V10AckSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for V10Ack class.

class Meta

Bases: object

V10Ack schema metadata.

model_class

alias of *aries_cloudagent.protocols.notification.v1_0.messages.ack.V10Ack*

status

Submodules

aries_cloudagent.protocols.notification.v1_0.message_types module

Message and inner object type identifiers for present-proof protocol v2.0.

Submodules

aries_cloudagent.protocols.notification.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.out_of_band package

Subpackages

aries_cloudagent.protocols.out_of_band.v1_0 package

Subpackages

aries_cloudagent.protocols.out_of_band.v1_0.handlers package

Submodules

aries_cloudagent.protocols.out_of_band.v1_0.handlers.problem_report_handler module

aries_cloudagent.protocols.out_of_band.v1_0.handlers.reuse_accept_handler module

aries_cloudagent.protocols.out_of_band.v1_0.handlers.reuse_handler module

aries_cloudagent.protocols.out_of_band.v1_0.messages package

Submodules

aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation module

An invitation content message.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.**HSProto**(*value*)

Bases: enum.Enum

Handshake protocol enum for invitation message.

RFC160 = HSProtoSpec(rfc=160, name='connections/1.0', aka={'160', 'connection', 'conn', 'connections', 'rfc160', 'old', 'conns'})

RFC23 = HSProtoSpec(rfc=23, name='didexchange/1.0', aka={'new', 'rfc23', 'didex', 'didexchange', 'didx', '23'})

property aka: int

Accessor for also-known-as.

classmethod get(*label: Union[str,*
aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.HSProto*]*) → *aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.HSProto*

Get handshake protocol enum for label.

property rfc: int
Accessor for RFC.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.HSProtoSpec(*rfc*, *name*, *aka*)

Bases: tuple

property aka
Alias for field number 2

property name
Alias for field number 1

property rfc
Alias for field number 0

class aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.**InvitationMessage**(*, *comment: Optional[str] = None*, *label: Optional[str] = None*, *image_url: Optional[str] = None*, *handshake_protocols: Optional[Sequence[str] = None*, *requests_attach: Optional[Sequence[ari = None*, *services: Optional[Sequence[Un str]]] = None*, *accept: Optional[Sequence[str] = None*, *version: str = '1.1'*, *msg_type: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing an out of band invitation message.

class Meta

Bases: object

InvitationMessage metadata.

message_type = 'out-of-band/1.1/invitation'

schema_class = 'InvitationMessageSchema'

classmethod from_url(*url: str*) → *aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.InvitationMessage*

Parse a URL-encoded invitation into an *InvitationMessage* instance.

Parameters **url** – Url to decode

Returns An *InvitationMessage* object.

to_url(*base_url: Optional[str] = None*) → str

Convert an invitation message to URL format for sharing.

Returns An invite url

classmethod wrap_message(*message: dict*) → *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*

Convert an aries message to an attachment decorator.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.**InvitationMessageSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

InvitationMessage schema.

class Meta

Bases: object

InvitationMessage schema metadata.

model_class

alias of *aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.InvitationMessage*

post_dump(*data*, ***kwargs*)

Post dump hook.

requests_attach

alias of *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecoratorSchema*

services

DIDComm Service object or DID string field for Marshmallow.

validate_fields(*data*, ***kwargs*)

Validate schema fields.

Parameters **data** – The data to validate

Raises **ValidationError** – If any of the fields do not validate

class aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.**ServiceOrDIDField**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.fields.

DIDComm Service object or DID string field for Marshmallow.

aries_cloudagent.protocols.out_of_band.v1_0.messages.problem_report module

Represents an OOB connection reuse problem report message.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.problem_report.**OOBProblemReport**(*version: str = '1.1'*, *msg_type: Optional[str] = None*, **args*, ***kwargs*)

Bases: *aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReport*

Base class representing an OOB connection reuse problem report message.

class Meta

Bases: object

OOB connection reuse problem report metadata.

handler_class = 'aries_cloudagent.protocols.out_of_band.v1_0.handlers.problem_report_handler.OOBProblemReportMessageHandler'

message_type = 'out-of-band/1.1/problem_report'

schema_class = 'OOBProblemReportSchema'

class aries_cloudagent.protocols.out_of_band.v1_0.messages.problem_report.**OOBProblemReportSchema**(**args: Any*, ***kwarg Any*)

Bases: marshmallow.

Schema for ProblemReport base class.

class Meta

Bases: object

Metadata for problem report schema.

model_class

alias of *aries_cloudagent.protocols.out_of_band.v1_0.messages.problem_report.OOBProblemReport*

check_thread_deco(*obj*, ***kwargs*)

Thread decorator, and its thid and pthid, are mandatory.

validate_fields(*data*, ***kwargs*)
Validate schema fields.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.problem_report.**ProblemReportReason**(*value*)
Bases: enum.Enum

Supported reason codes.

EXISTING_CONNECTION_NOT_ACTIVE = 'existing_connection_not_active'

NO_EXISTING_CONNECTION = 'no_existing_connection'

aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse module

Represents a Handshake Reuse message under RFC 0434.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse.**HandshakeReuse**(*version: str = '1.1'*, *msg_type: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a Handshake Reuse message.

class Meta
Bases: object

Metadata for Handshake Reuse message.

handler_class = 'aries_cloudagent.protocols.out_of_band.v1_0.handlers.reuse_handler.HandshakeReuseMessageHandler'

message_type = 'out-of-band/1.1/handshake-reuse'

schema_class = 'HandshakeReuseSchema'

class aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse.**HandshakeReuseSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Handshake Reuse schema class.

class Meta
Bases: object

Handshake Reuse schema metadata.

model_class
alias of *aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse.HandshakeReuse*

check_thread_deco(*obj*, ***kwargs*)
Thread decorator, and its thid and pthid, are mandatory.

aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse_accept module

Represents a Handshake Reuse Accept message under RFC 0434.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse_accept.**HandshakeReuseAccept**(*version: str = '1.1', msg_type: Optional[str] = None, **kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a Handshake Reuse Accept message.

class Meta

Bases: object

Metadata for Handshake Reuse Accept message.

handler_class = 'aries_cloudagent.protocols.out_of_band.v1_0.handlers.reuse_accept_handler.HandshakeReuseAcceptMessageHandler'

message_type = 'out-of-band/1.1/handshake-reuse-accepted'

schema_class = 'HandshakeReuseAcceptSchema'

class aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse_accept.**HandshakeReuseAcceptSchema**(**args Any, **kw Any*)

Bases: marshmallow.

Handshake Reuse Accept schema class.

class Meta

Bases: object

Handshake Reuse Accept schema metadata.

model_class

alias of *aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse_accept.HandshakeReuseAccept*

check_thread_deco(*obj*, ***kwargs*)

Thread decorator, and its thid and pthid, are mandatory.

aries_cloudagent.protocols.out_of_band.v1_0.messages.service module

Record used to represent a service block of an out of band invitation.

class aries_cloudagent.protocols.out_of_band.v1_0.messages.service.**Service**(*, *_id: Optional[str] = None*, *_type: Optional[str] = None*, *did: Optional[str] = None*, *recipient_keys: Optional[Sequence[str]] = None*, *routing_keys: Optional[Sequence[str]] = None*, *service_endpoint: Optional[str] = None*)

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

Record used to represent a service block of an out of band invitation.

class Meta

Bases: object

Service metadata.

schema_class = 'ServiceSchema'

class aries_cloudagent.protocols.out_of_band.v1_0.messages.service.**ServiceSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Service schema.

class Meta

Bases: object

ServiceSchema metadata.

model_class

alias of *aries_cloudagent.protocols.out_of_band.v1_0.messages.service.Service*

post_dump(*data*, ***kwargs*)

Post dump hook.

aries_cloudagent.protocols.out_of_band.v1_0.models package

Submodules

aries_cloudagent.protocols.out_of_band.v1_0.models.invitation module

Record for out of band invitations.

class aries_cloudagent.protocols.out_of_band.v1_0.models.invitation.**InvitationRecord**(*, *invitation_id: Optional[str] = None*, *state: Optional[str] = None*, *invi_msg_id: Optional[str] = None*, *invitation: Optional[Union[aries_clou Mapping]] = None*, *invitation_url: Optional[str] = None*, *oob_id: Optional[str] = None*, *public_did: Optional[str] = None*, *trace: bool = False*, ***kwargs*)

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

Represents an out of band invitation record.

class Meta

Bases: object

InvitationRecord metadata.

schema_class = 'InvitationRecordSchema'

RECORD_ID_NAME = 'invitation_id'

RECORD_TOPIC: Optional[str] = 'oob_invitation'

RECORD_TYPE = 'oob_invitation'

STATE_AWAIT_RESPONSE = 'await_response'

STATE_DONE = 'done'

STATE_INITIAL = 'initial'

TAG_NAMES = {'invi_msg_id'}

property invitation:
aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.InvitationMessage
Accessor; get deserialized view.

property invitation_id: str
Accessor for the ID associated with this exchange.

property record_value: dict
Accessor for the JSON record value generated for this invitation.

class aries_cloudagent.protocols.out_of_band.v1_0.models.invitation.**InvitationRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema to allow serialization/deserialization of invitation records.

class Meta
Bases: object

InvitationRecordSchema metadata.

model_class
alias of *aries_cloudagent.protocols.out_of_band.v1_0.models.invitation.InvitationRecord*

invi_msg_id

invitation

invitation_id

invitation_url

oob_id

state

aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record module

Record for out of band invitations.

class aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record.**OobRecord**(*, *state: str*, *invi_msg_id: str*, *role: str*, *invitation: Union[*aries_cloudagent.protocol *Mapping[str, Any]]*, *their_service: Optional[*aries_cloudagent.messagi = *None*, *connection_id: Optional[str]* = *None*, *reuse_msg_id: Optional[str]* = *None*, *oob_id: Optional[str]* = *None*, *attach_thread_id: Optional[str]* = *None*, *our_recipient_key: Optional[str]* = *None*, *our_service: Optional[*aries_cloudagent.messagi = *None*, *multi_use: bool* = *False*, *trace: bool* = *False*, ***kwargs*)

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

Represents an out of band record.

class **Meta**

Bases: object

OobRecord metadata.

schema_class = 'OobRecordSchema'

RECORD_ID_NAME = 'oob_id'

RECORD_TOPIC: Optional[str] = 'out_of_band'

RECORD_TYPE = 'oob_record'

RECORD_TYPE_METADATA = 'connection_metadata'

ROLE_RECEIVER = 'receiver'

ROLE_SENDER = 'sender'

STATE_ACCEPTED = 'reuse-accepted'

STATE_AWAIT_RESPONSE = 'await-response'

STATE_DONE = 'done'

STATE_INITIAL = 'initial'

STATE_NOT_ACCEPTED = 'reuse-not-accepted'

STATE_PREPARE_RESPONSE = 'prepare-response'

TAG_NAMES = {'attach_thread_id', 'connection_id', 'invi_msg_id', 'our_recipient_key', 'reuse_msg_id'}

async delete_record(*session:* aries_cloudagent.core.profile.ProfileSession)
Perform connection record deletion actions.

Parameters **session** (*ProfileSession*) – session

property invitation:
aries_cloudagent.protocols.out_of_band.v1_0.messages.invitation.InvitationMessage
Accessor; get deserialized view.

async metadata_delete(*session:* aries_cloudagent.core.profile.ProfileSession, *key: str*)
Delete custom metadata associated with this connection.

Parameters

- **session** (*ProfileSession*) – session used for storage
- **key** (*str*) – key of metadata to delete

async metadata_get(*session:* aries_cloudagent.core.profile.ProfileSession, *key: str*, *default: Optional[Any] = None*) → Any
Retrieve arbitrary metadata associated with this connection.

Parameters

- **session** (*ProfileSession*) – session used for storage
- **key** (*str*) – key identifying metadata
- **default** (*Any*) – default value to get; type should be a JSON compatible value.

Returns metadata stored by key

Return type Any

async metadata_get_all(*session:* aries_cloudagent.core.profile.ProfileSession) → dict
Return all custom metadata associated with this connection.

Parameters **session** (*ProfileSession*) – session used for storage

Returns dictionary representation of all metadata values

Return type dict

async metadata_set(*session:* aries_cloudagent.core.profile.ProfileSession, *key: str*, *value: Any*)
Set arbitrary metadata associated with this connection.

Parameters

- **session** (*ProfileSession*) – session used for storage
- **key** (*str*) – key identifying metadata

- **value** (*Any*) – value to set

property oob_id: str
Accessor for the ID associated with this exchange.

property record_value: dict
Accessor for the JSON record value generated for this invitation.

class aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record.**OobRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema to allow serialization/deserialization of invitation records.

class Meta
Bases: object

OobRecordSchema metadata.

model_class
alias of *aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record.OobRecord*

attach_thread_id

connection_id

invi_msg_id

invitation

oob_id

our_recipient_key

role

state

their_service

Submodules

aries_cloudagent.protocols.out_of_band.v1_0.controller module

Protocol controller for out-of-band.

class aries_cloudagent.protocols.out_of_band.v1_0.controller.**Controller**(*protocol: str*)
Bases: object

Out-of-band protocol controller.

determine_goal_codes() → Sequence[str]
Return defined goal_codes.

aries_cloudagent.protocols.out_of_band.v1_0.manager module

aries_cloudagent.protocols.out_of_band.v1_0.message_types module

Message and inner object type identifiers for Out of Band messages.

aries_cloudagent.protocols.out_of_band.v1_0.routes module

Submodules

aries_cloudagent.protocols.out_of_band.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.present_proof package

Subpackages

aries_cloudagent.protocols.present_proof.dif package

Submodules

aries_cloudagent.protocols.present_proof.dif.pres_exch module

Schemas for dif presentation exchange attachment.

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**ClaimFormat**(*, *jwt: Optional[Mapping] = None*, *jwt_vc: Optional[Mapping] = None*, *jwt_vp: Optional[Mapping] = None*, *ldp: Optional[Mapping] = None*, *ldp_vc: Optional[Mapping] = None*, *ldp_vp: Optional[Mapping] = None*, *Optional[Mapping] = None*)

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

Defines Claim field.

class Meta

Bases: object

ClaimFormat metadata.

schema_class = 'ClaimFormatSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**ClaimFormatSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single ClaimFormat Schema.

class Meta

Bases: object

ClaimFormatSchema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.ClaimFormat*

jwt

jwt_vc

jwt_vp

ldp

ldp_vc

ldp_vp

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**Constraints**(***, *subject_issuer: Optional[str] = None*, *limit_disclosure: Optional[bool] = None*, *holders: Optional[Sequence[*aries_cloudagent.proto *= None*, *_fields: Optional[Sequence[*aries_cloudagent.proto *= None*, *status_active: Optional[str] = None*, *status_suspended: Optional[str] = None*, *status_revoked: Optional[str] = None*)

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

Single Constraints which describes InputDescriptor's Contraint field.

class Meta

Bases: object

Constraints metadata.

schema_class = 'ConstraintsSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**ConstraintsSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single Constraints Schema.

class Meta
Bases: object

ConstraintsSchema metadata.

model_class
alias of aries_cloudagent.protocols.present_proof.dif.pres_exch.Constraints

extract_info(*data*, ***kwargs*)
Support deserialization of statuses according to DIF spec.

holders

limit_disclosure

reformat_data(*data*, ***kwargs*)
Support serialization of statuses according to DIF spec.

status_active

status_revoked

status_suspended

subject_issuer

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**DIFField**(*, *id: Optional[str] = None*, *paths: Optional[Sequence[str]] = None*, *purpose: Optional[str] = None*, *predicate: Optional[str] = None*, *_filter: Optional[*aries_cloudagent.protocols.present_p *= None*)

Bases: aries_cloudagent.messaging.models.base.BaseModel

Single Field object for the Constraint.

class Meta
Bases: object

Field metadata.

schema_class = 'DIFFieldSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**DIFFieldSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single Field Schema.

class Meta
Bases: object

DIFFieldSchema metadata.

model_class
alias of aries_cloudagent.protocols.present_proof.dif.pres_exch.DIFField

id

paths

predicate

purpose

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**DIFHolder**(*, *field_ids: Optional[Sequence[str]] = None*, *directive: Optional[str] = None*)

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

Single Holder object for Constraints.

class Meta

Bases: object

Holder metadata.

schema_class = 'DIFHolderSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**DIFHolderSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single Holder Schema.

class Meta

Bases: object

DIFHolderSchema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.DIFHolder*

directive

field_ids

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**DIFOptions**(*, *challenge: Optional[str] = None*, *domain: Optional[str] = None*)

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

Single DIFOptions object.

class Meta

Bases: object

DIFOptions metadata.

schema_class = 'DIFOptionsSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**DIFOptionsSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for options required for the Prover to fulfill the Verifier's request.

class Meta

Bases: object

DIFOptionsSchema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.DIFOptions*

challenge

domain

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**Filter**(*, *_not: bool = False*, *_type: Optional[str] = None*, *fmt: Optional[str] = None*, *pattern: Optional[str] = None*, *minimum: Optional[str] = None*, *maximum: Optional[str] = None*, *min_length: Optional[int] = None*, *max_length: Optional[int] = None*, *exclusive_min: Optional[str] = None*, *exclusive_max: Optional[str] = None*, *const: Optional[str] = None*, *enums: Optional[Sequence[str]] = None*)

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

Single Filter for the Constraint object.

class Meta
Bases: object

Filter metadata.

schema_class = 'FilterSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**FilterSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single Filter Schema.

class Meta
Bases: object

FilterSchema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter*

const
String or Number field for Marshmallow.

enums

exclusive_max
String or Number field for Marshmallow.

exclusive_min
String or Number field for Marshmallow.

extract_info(*data*, ***kwargs*)
Enum validation and not filter logic.

fmt

max_length

maximum
String or Number field for Marshmallow.

min_length

minimum
String or Number field for Marshmallow.

pattern

serialize_reformat(*data*, ***kwargs*)
Support serialization of not filter according to DIF spec.

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**InputDescriptorMapping**(*, *id: Optional[str] = None*, *fmt: Optional[str] = None*, *path: Optional[str] = None*)

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

Single InputDescriptorMapping object.

class Meta
Bases: object

InputDescriptorMapping metadata.

schema_class = 'InputDescriptorMappingSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**InputDescriptorMappingSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single InputDescriptorMapping Schema.

class Meta
Bases: object

InputDescriptorMappingSchema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.InputDescriptorMapping*

fmt

id

path

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**InputDescriptors**(*, *id: Optional[str] = None*, *groups: Optional[Sequence[str]] = None*, *name: Optional[str] = None*, *purpose: Optional[str] = None*, *metadata: Optional[dict] = None*, *constraint: Optional[*aries_cloudagent.protocols *= None*, *schemas: Optional[*aries_cloudagent.protocols *= None*)

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

Input Descriptors.

class Meta

Bases: object

InputDescriptors metadata.

schema_class = 'InputDescriptorsSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**InputDescriptorsSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single InputDescriptors Schema.

class Meta

Bases: object

InputDescriptorsSchema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.InputDescriptors*

constraint

groups

id

metadata

name

purpose

schemas

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**PresentationDefinition**(*, *id: Optional[str] = None*, *name: Optional[str] = None*, *purpose: Optional[str] = None*, *fmt: Optional[aries_cloudagent.p = None*, *submission_requirements: Optional[Sequence[aries_cl = None*, *input_descriptors: Optional[Sequence[aries_cl = None*, ***kwargs*)

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

https://identity.foundation/presentation-exchange/.

class Meta

Bases: object

PresentationDefinition metadata.

schema_class = 'PresentationDefinitionSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**PresentationDefinitionSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single Presentation Definition Schema.

class Meta

Bases: object

PresentationDefinitionSchema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.PresentationDefinition*

fmt

id

input_descriptors

name

purpose

submission_requirements

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**PresentationSubmission**(*, *id: Optional[str] = None*, *definition_id: Optional[str] = None*, *descriptor_maps: Optional[Sequence[*aries_cl
= None)

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

Single PresentationSubmission object.

class Meta

Bases: object

PresentationSubmission metadata.

schema_class = 'PresentationSubmissionSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**PresentationSubmissionSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single PresentationSubmission Schema.

class Meta

Bases: object

PresentationSubmissionSchema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.PresentationSubmission*

definition_id

descriptor_maps

id

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**Requirement**(*, *count: Optional[int] = None*, *maximum: Optional[int] = None*, *minimum: Optional[int] = None*, *input_descriptors: Optional[Sequence[*aries_cloudagent.proto *= None*, *nested_req: Optional[Sequence] = None*)

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

Single Requirement generated from toRequirement function.

class Meta
Bases: object

Requirement metadata.

schema_class = 'RequirementSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**RequirementSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single Requirement Schema.

class Meta
Bases: object

RequirementSchema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.Requirement*

count

input_descriptors

maximum

minimum

nested_req

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**SchemaInputDescriptor**(*, *uri: Optional[str] = None*, *required: Optional[bool] = None*)

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

SchemaInputDescriptor.

class Meta

Bases: object

SchemaInputDescriptor metadata.

schema_class = 'SchemaInputDescriptorSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**SchemaInputDescriptorSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single SchemaField Schema.

class Meta

Bases: object

SchemaInputDescriptorSchema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.SchemaInputDescriptor*

required

uri

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**SchemasInputDescriptorFilter**(*, *oneof_filter: bool = False*, *uri_groups: Optional[Sequence[= None*)

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

SchemasInputDescriptorFilter.

class Meta

Bases: object

InputDescriptor Schemas filter metadata.

schema_class = 'SchemasInputDescriptorFilterSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**SchemasInputDescriptorFilterSchema**(**args: Any*, ***kwargs Any*)

Bases: marshmallow.

Single SchemasInputDescriptorFilterSchema Schema.

class Meta

Bases: object

SchemasInputDescriptorFilterSchema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.SchemasInputDescriptorFilter*

extract_info(*data*, ***kwargs*)

deserialize.

oneof_filter

uri_groups

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**SubmissionRequirements**(*, *_name: Optional[str] = None*, *purpose: Optional[str] = None*, *rule: Optional[str] = None*, *count: Optional[int] = None*, *minimum: Optional[int] = None*, *maximum: Optional[int] = None*, *_from: Optional[str] = None*, *from_nested: Optional[Sequence] = None*)

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

describes input to be submitted via a presentation submission.

class Meta

Bases: object

SubmissionRequirements metadata.

schema_class = 'SubmissionRequirementsSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**SubmissionRequirementsSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Single Presentation Definition Schema.

class Meta

Bases: object

SubmissionRequirementsSchema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.SubmissionRequirements*

count

from_nested

maximum

minimum

purpose

rule

validate_from(*data*, ***kwargs*)

Support validation of from and from_nested.

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**VerifiablePresentation**(*, *id: Optional[str] = None, contexts: Optional[Sequence[Union[s dict]]] = None, types: Optional[Sequence[str]] = None, credentials: Optional[Sequence[dict]] = None, proof: Optional[Sequence[dict]] = None, presentation_submission: Optional[aries_cloudagent.p = None*)

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

Single VerifiablePresentation object.

class Meta

Bases: object

VerifiablePresentation metadata.

schema_class = 'VerifiablePresentationSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_exch.**VerifiablePresentationSchema**(**args: Any, **kwargs: Any*)

Bases: marshmallow.

Single Verifiable Presentation Schema.

class Meta

Bases: object

VerifiablePresentationSchema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.dif.pres_exch.VerifiablePresentation*

contexts

credentials

id

presentation_submission

proof

types

aries_cloudagent.protocols.present_proof.dif.pres_exch_handler module

Utilities for dif presentation exchange attachment.

General Flow: create_vp -> make_requirement [create a Requirement from SubmissionRequirements and Descriptors] -> apply_requirement [filter credentials] -> merge [return applicable credential list and descriptor_map for presentation_submission] returns VerifiablePresentation

exception aries_cloudagent.protocols.present_proof.dif.pres_exch_handler.**DIFPresExchError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base class for DIF Presentation Exchange related errors.

class aries_cloudagent.protocols.present_proof.dif.pres_exch_handler.**DIFPresExchHandler**(*profile: aries_cloudagent.co*, *pres_signing_did: Optional[str] = None*, *proof_type: Optional[str] = None*, *reveal_doc: Optional[dict] = None*)

Bases: `object`

Base Presentation Exchange Handler.

DERIVED_PROOF_TYPE_SIGNATURE_SUITE_MAPPING = {'BbsBlsSignatureProof2020': <class 'aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_proof_2020.BbsBlsSignatureProof2020'>}

DERIVE_SIGNATURE_SUITE_KEY_TYPE_MAPPING = {<class 'aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_proof_2020.BbsBlsSignatureProof2020'>: <aries_cloudagent.wallet.key_type.KeyType object>}

ISSUE_SIGNATURE_SUITE_KEY_TYPE_MAPPING = {<class 'aries_cloudagent.vc.ld_proofs.suites.ed25519_signature_2018.Ed25519Signature2018'>: <aries_cloudagent.wallet.key_type.KeyType object>}

PROOF_TYPE_SIGNATURE_SUITE_MAPPING = {'Ed25519Signature2018': <class 'aries_cloudagent.vc.ld_proofs.suites.ed25519_signature_2018.Ed25519Signature2018'>}

async apply_constraint_received_cred(*constraint:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Constraints, *cred_dict: dict*) → bool

Evaluate constraint from the request against received credential.

async apply_requirements(*req:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Requirement, *credentials: Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]*, *records_filter: Optional[dict] = None*) → dict

Apply Requirement.

Parameters

- **req** – Requirement
- **credentials** – Sequence of credentials to check against

Returns dict of input_descriptor ID key to list of credential_json

build_nested_paths_dict(*key: str*, *value: str*, *nested_field_paths: dict*, *cred_dict: dict*) → dict

Build and return nested_field_paths dict.

check_attr_in_extracted_dict(*extracted_dict: dict*, *nested_attr_values: dict*) → bool

Check if keys of extracted_dict exists in nested_attr_values.

check_filter_only_type_enforced(*_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool

Check if only type is specified in filter.

Parameters **_filter** – Filter

Returns bool

check_if_cred_id_derived(*id: str*) → bool

Check if credential or credentialSubjet id is derived.

const_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool

Const check.

Returns True if value is equal to filter specified check

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

contains(*data: Sequence[str]*, *e: str*) → bool
Check for e in data.

Returns True if e exists in data else return False

Parameters

- **data** – Sequence of str
- **e** – str value to check

Returns bool

create_vcrecord(*cred_dict: dict*) → *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*
Return VCRecord from a credential dict.

async create_vp(*credentials: Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]*, *pd:* aries_cloudagent.protocols.present_proof.dif.pres_exch.PresentationDefinition, *challenge: Optional[str] = None*, *domain: Optional[str] = None*, *records_filter: Optional[dict] = None*) → Union[Sequence[dict], dict]
Create VerifiablePresentation.

Parameters

- **credentials** – Sequence of VCRecords
- **pd** – PresentationDefinition

Returns VerifiablePresentation

credential_match_schema(*credential:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord, *schema_id: str*) → bool
Credential matching by schema.

Used by filter_schema to check if credential.schema_ids or credential.types matched with schema_id

Parameters

- **credential** – VCRecord to check
- **schema_id** – schema uri to check

Returns bool

enum_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Enum check.

Returns True if value is contained to filter specified list

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

exclusive_maximum_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Exclusivemaximum check.

Returns True if value less than filter specified check

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

exclusive_minimum_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool

Exclusiveminimum check.

Returns True if value greater than filter specified check

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

field_ids_for_is_holder(*constraints:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Constraints) → Sequence[str]

Return list of field ids for whose subject holder verification is requested.

async filter_by_field(*field:* aries_cloudagent.protocols.present_proof.dif.pres_exch.DIFField, *credential:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord) → bool

Apply filter on VCRecord.

Checks if a credential is applicable

Parameters

- **field** – Field contains filtering spec
- **credential** – credential to apply filtering on

Returns bool

async filter_constraints(*constraints:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Constraints, *credentials: Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]*) → Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]

Return list of applicable VCRecords after applying filtering.

Parameters

- **constraints** – Constraints
- **credentials** – Sequence of credentials to apply filtering on

Returns Sequence of applicable VCRecords

async filter_creds_record_id(*credentials: Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]*, *records_list: Sequence[str]*) → Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]

Return filtered list of credentials using records_list.

async filter_schema(*credentials: Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]*, *schemas:* aries_cloudagent.protocols.present_proof.dif.pres_exch.SchemasInputDescriptorFilter) → Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]

Filter by schema.

Returns list of credentials where credentialSchema.id or types matched with input_descriptors.schema.uri

Parameters

- **credentials** – list of VCRecords to check
- **schemas** – list of schemas from the input_descriptors

Returns Sequence of filtered VCRecord

get_dict_keys_from_path(*derived_cred_dict: dict*, *path: str*) → List
Return additional_attrs to build nested_field_paths.

async get_sign_key_credential_subject_id(*applicable_creds: Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]*) → Tuple[Optional[str], Sequence[dict]]
Get the issuer_id and filtered_creds from enclosed credentials subject_ids.

async get_updated_field(*field:* aries_cloudagent.protocols.present_proof.dif.pres_exch.DIFField, *cred: dict*) → *aries_cloudagent.protocols.present_proof.dif.pres_exch.DIFField*
Return field with updated json path, if necessary.

async get_updated_path(*cred_dict: dict*, *json_path: str*) → str
Return updated json path, if necessary.

is_len_applicable(*req:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Requirement, *val: int*) → bool
Check and validate requirement minimum, maximum and count.

Parameters

- **req** – Requirement
- **val** – int value to check

Returns bool

is_numeric(*val: any*)
Check if val is an int or float.

Parameters **val** – to check

Returns numeric value

Raises ***DIFPresExchError*** – Provided value has invalid/incompatible type

length_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Length check.

Returns True if length value string meets the minLength and maxLength specs

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

async make_requirement(*srs: Optional[Sequence[*aries_cloudagent.protocols.present_proof.dif.pres_exch.SubmissionRequirements*]] = None*, *descriptors: Optional[Sequence[*aries_cloudagent.protocols.present_proof.dif.pres_exch.InputDescriptors*]] = None*) → *aries_cloudagent.protocols.present_proof.dif.pres_exch.Requirement*
Return Requirement.

Creates and return Requirement with nesting if required using to_requirement()

Parameters

- **srs** – list of submission_requirements
- **descriptors** – list of input_descriptors

Raises ***DIFPresExchError*** – If not able to create requirement

maximum_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Maximum check.

Returns True if value less than equal to filter specified check

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

async merge(*dict_descriptor_creds: dict*) →
Tuple[Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*],
Sequence[*aries_cloudagent.protocols.present_proof.dif.pres_exch.InputDescriptorMapping*]]
Return applicable credentials and descriptor_map for attachment.

Used for generating the presentation_submission property with the descriptor_map, mantaining the order in which applicable credential list is returned.

Parameters

- **dict_descriptor_creds** – dict with input_descriptor.id as keys
- **merged_credentials_list** (*and*) –

Returns Tuple of applicable credential list and descriptor map

async merge_nested_results(*nested_result: Sequence[dict]*, *exclude: dict*) → dict
Merge nested results with merged credentials.

Parameters

- **nested_result** – Sequence of dict containing input_descriptor.id as keys and list of creds as values
- **exclude** – dict containing info about credentials to exclude

Returns dict with input_descriptor.id as keys and merged_credentials_list as values

minimum_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Minimum check.

Returns True if value greater than equal to filter specified check

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

nested_get(*input_dict: dict*, *path: str*) → Union[Dict, List]
Return dict or list from nested dict given list of nested_key.

new_credential_builder(*new_credential: dict*, *unflatten_dict: dict*) → dict
Update and return the new_credential.

Parameters

- **new_credential** – credential dict to be updated and returned
- **unflatten_dict** – dict with traversal path as key and match_value as value

Returns dict

pattern_check(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Pattern check.

Returns True if value string matches the specified pattern

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

async process_constraint_holders(*subject_ids: Sequence[str]*) → bool
Check if holder or subject of claim still controls the identifier.

process_numeric_val(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Trigger Filter checks.

Trigger appropriate check for a number type filter, according to _filter spec.

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

process_string_val(*val: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool
Trigger Filter checks.

Trigger appropriate check for a string type filter, according to _filter spec.

Parameters

- **val** – value to check, extracted from match
- **_filter** – Filter

Returns bool

async restrict_field_paths_one_of_filter(*field_paths: Sequence[str]*, *cred_dict: dict*) → Sequence[str]
Return field_paths that are applicable to oneof_filter.

reveal_doc(*credential_dict: dict*, *constraints:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Constraints)
Generate reveal_doc dict for deriving credential.

string_to_timezone_aware_datetime(*datetime_str: str*) → datetime.datetime
Convert string with PYTZ timezone to datetime for comparison.

subject_is_issuer(*credential:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord) → bool
subject_is_issuer check.

Returns True if cred issuer_id is in subject_ids

Parameters **credential** – VCRecord

Returns bool

async to_requirement(*sr:* aries_cloudagent.protocols.present_proof.dif.pres_exch.SubmissionRequirements, *descriptors: Sequence[*aries_cloudagent.protocols.present_proof.dif.pres_exch.InputDescriptors*]*) → *aries_cloudagent.protocols.present_proof.dif.pres_exch.Requirement*

Return Requirement.

Parameters

- **sr** – submission_requirement
- **descriptors** – list of input_descriptors

Raises ***DIFPresExchError*** – If not able to create requirement

validate_patch(*to_check: any*, *_filter:* aries_cloudagent.protocols.present_proof.dif.pres_exch.Filter) → bool

Apply filter on match_value.

Utility function used in applying filtering to a cred by triggering checks according to filter specification

Parameters

- **to_check** – value to check, extracted from match
- **_filter** – Filter

Returns bool

async verify_received_pres(*pd:* aries_cloudagent.protocols.present_proof.dif.pres_exch.PresentationDefinition, *pres: Union[Sequence[dict], dict]*)

Verify credentials received in presentation.

Parameters

- **pres** – received VerifiablePresentation
- **pd** – PresentationDefinition

aries_cloudagent.protocols.present_proof.dif.pres_proposal_schema module

DIF Proof Proposal Schema.

class aries_cloudagent.protocols.present_proof.dif.pres_proposal_schema.**DIFProofProposalSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for DIF Proposal.

input_descriptors

options

aries_cloudagent.protocols.present_proof.dif.pres_request_schema module

DIF Proof Request Schema.

class aries_cloudagent.protocols.present_proof.dif.pres_request_schema.**DIFPresSpecSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for DIF Presentation Spec schema.

issuer_id

presentation_definition

record_ids

reveal_doc

class aries_cloudagent.protocols.present_proof.dif.pres_request_schema.**DIFProofRequest**(*presentation_definitio* *Optional[Union[dict,* aries_cloudagent.prot *options:* *Optional[Union[dict,* aries_cloudagent.prot *= None*)

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

DIF presentation request input detail.

class Meta

Bases: object

DIFProofRequest metadata.

schema_class = 'DIFProofRequestSchema'

class aries_cloudagent.protocols.present_proof.dif.pres_request_schema.**DIFProofRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for DIF presentation request.

class Meta

Bases: object

Accept parameter overload.

model_class

alias of *aries_cloudagent.protocols.present_proof.dif.pres_request_schema.DIFProofRequest*

options

presentation_definition

aries_cloudagent.protocols.present_proof.dif.pres_schema module

DIF Proof Schema.

class aries_cloudagent.protocols.present_proof.dif.pres_schema.**DIFProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for DIF Proof.

contexts

credentials

id

presentation_submission

proof

types

aries_cloudagent.protocols.present_proof.indy package

Submodules

aries_cloudagent.protocols.present_proof.indy.pres_exch_handler module

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

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_handler module

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_problem_report_handler module

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_proposal_handler module

aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_request_handler module

aries_cloudagent.protocols.present_proof.v1_0.messages package

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: Optional[str] = None*, ***, *comment: Optional[str] = None*, *presentations_attach: Optional[Sequence[aries = None*, ***kwargs*)

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.presentation_handler.PresentationHandler'

message_type = '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.**PresentationSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

(Proof) presentation schema.

class Meta

Bases: object

Presentation schema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.v1_0.messages.presentation.Presentation*

comment

presentations_attach

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.**PresentationAck**(*status: Optional[str] = None*, *verification_result: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.protocols.notification.v1_0.messages.ack.V10Ack*

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.presentation_ack_handler.PresentationAckHandler'

message_type = 'present-proof/1.0/ack'

schema_class = 'PresentationAckSchema'

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_ack.**PresentationAckSchema**(**arg Any, **kw Any*)

Bases: marshmallow.

Schema for PresentationAck class.

class Meta
Bases: object

PresentationAck schema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_ack.PresentationAck*

verification_result

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_problem_report module

A problem report message.

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_problem_report.**PresentationPr**

Bases: *aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReport*

Class representing a problem report message.

class Meta

Bases: object

Problem report metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v1_0.handlers.presentation_problem_report_handler.PresentationProblemReportHandler'

message_type = 'present-proof/1.0/problem-report'

schema_class = 'PresentationProblemReportSchema'

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_problem_report.**PresentationPr**

Bases: marshmallow.

Problem report schema.

class Meta

Bases: object

Schema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_problem_report.PresentationProblemReport*

validate_fields(*data*, ***kwargs*)

Validate schema fields.

Parameters **data** – The data to validate

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_problem_report.**ProblemReportR**

Bases: enum.Enum

Supported reason codes.

ABANDONED = 'abandoned'

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_proposal module

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.**PresentationRequest**(_

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.presentation_request_handler.PresentationRequestHandler'

message_type = 'present-proof/1.0/request-presentation'

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.**PresentationRequestSch**

Bases: marshmallow.

Presentation request schema.

class Meta

Bases: object

Presentation request schema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_request.PresentationRequest*

comment

request_presentations_attach

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_webhook module

v1.0 presentation exchange information webhook.

class aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_webhook.**V10PresentationExchang**
Bases: object

Class representing a state only presentation exchange webhook.

aries_cloudagent.protocols.present_proof.v1_0.models package

Package-wide data and code.

Submodules

aries_cloudagent.protocols.present_proof.v1_0.models.presentation_exchange module

Submodules

aries_cloudagent.protocols.present_proof.v1_0.controller module

aries_cloudagent.protocols.present_proof.v1_0.manager module

aries_cloudagent.protocols.present_proof.v1_0.message_types module

Message and inner object type identifiers for present-proof protocol v1.0.

aries_cloudagent.protocols.present_proof.v1_0.routes module

aries_cloudagent.protocols.present_proof.v2_0 package

aries_cloudagent.protocols.present_proof.v2_0.**problem_report_for_record**(*record: Union[*aries_cloudagent.connections.mod
aries_cloudagent.protocols.present_proof
desc_en: str) → *aries_cloudagent.protocols.present_proo*

Create problem report for record.

Parameters

- **record** – connection or exchange record
- **desc_en** – description text to include in problem report

async aries_cloudagent.protocols.present_proof.v2_0.**report_problem**(*err:* aries_cloudagent.core.error.BaseError, *desc_en: str*, *http_error_class*, *record:* *Union[*aries_cloudagent.connections.models.co aries_cloudagent.protocols.present_proof.v2_0. *outbound_handler:* *Coroutine*)

Send problem report response and raise corresponding HTTP error.

Parameters

- **err** – error for internal diagnostics
- **desc_en** – description text to include in problem report (response)
- **http_error_class** – HTTP error to raise
- **record** – record to cite by thread in problem report
- **outbound_handler** – outbound message handler

Subpackages

aries_cloudagent.protocols.present_proof.v2_0.formats package

Subpackages

aries_cloudagent.protocols.present_proof.v2_0.formats.dif package

Submodules

aries_cloudagent.protocols.present_proof.v2_0.formats.dif.handler module

V2.0 present-proof dif presentation-exchange format handler.

class aries_cloudagent.protocols.present_proof.v2_0.formats.dif.handler.**DIFPresFormatHandler**(*profile:* aries_cloudag

Bases: *aries_cloudagent.protocols.present_proof.v2_0.formats.handler.V20PresFormatHandler*

DIF presentation format handler.

ISSUE_SIGNATURE_SUITE_KEY_TYPE_MAPPING = {<class 'aries_cloudagent.vc.ld_proofs.suites.ed25519_signature_2018.Ed25519Signature2018'>: <aries_cloudagent.wallet.key_type.KeyType object>}

async **create_bound_request**(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord, *request_data: Optional[dict] = None*) → Tuple[*aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create a presentation request bound to a proposal.

Parameters

- **pres_ex_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)

async create_pres(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord, *request_data: dict = {}*) → Tuple[*aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create a presentation.

format: *aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat.Format* = FormatSpec(aries='dif/', handler=<aries_cloudagent.utils.classloader.DeferLoad object>)

get_format_data(*message_type: str*, *data: dict*) → Tuple[*aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Get presentation format and attach objects for use in pres_ex messages.

get_format_identifier(*message_type: str*) → str

Get attachment format identifier for format and message combination.

Parameters **message_type** (*str*) – Message type for which to return the format identifier

Returns Issue credential attachment format identifier

Return type str

async process_vcrecords_return_list(*vc_records: Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]*, *record_ids: set*) → Tuple[Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*], set]

Return list of non-duplicate VCRecords.

async receive_pres(*message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres.V20Pres, *pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord)

Receive a presentation, from message in context on manager creation.

async retrieve_uri_list_from_schema_filter(*schema_uri_groups: Sequence[Sequence[*aries_cloudagent.protocols.present_proof.dif.pres_exch
→ Sequence[str]

Retrieve list of schema uri from uri_group.

classmethod validate_fields(*message_type: str*, *attachment_data: Mapping*)

Validate attachment data for a specific message type.

Uses marshmallow schemas to validate if format specific attachment data is valid for the specified message type. Only does structural and type checks, does not validate if .e.g. the issuer value is valid.

Parameters

- **message_type** (*str*) – The message type to validate the attachment data for. Should be one of the message types as defined in message_types.py

- **attachment_data** (*Mapping*) – [description] The attachment data to valide

Raises **Exception** – When the data is not valid.

async verify_pres(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord) → *aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord*

Verify a presentation.

Parameters **pres_ex_record** – presentation exchange record with presentation request and presentation to verify

Returns presentation exchange record, updated

aries_cloudagent.protocols.present_proof.v2_0.formats.indy package

Submodules

aries_cloudagent.protocols.present_proof.v2_0.formats.indy.handler module

Submodules

aries_cloudagent.protocols.present_proof.v2_0.formats.handler module

present-proof-v2 format handler - supports DIF and INDY.

class aries_cloudagent.protocols.present_proof.v2_0.formats.handler.**V20PresFormatHandler**(*profile:* aries_cloudagent.c

Bases: abc.ABC

Base Presentation Exchange Handler.

abstract async create_bound_request(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresEx *request_data: Optional[dict] = None*) → Tuple[*aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20Pre* *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create a presentation request bound to a proposal.

abstract async create_pres(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord, *request_data: Optional[dict] = None*) → Tuple[*aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Create a presentation.

format: *aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat.Format* = **None**

abstract get_format_data(*message_type: str*, *data: dict*) → Tuple[*aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat*, *aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator*]

Get presentation format and attach objects for use in pres_ex messages.

abstract get_format_identifier(*message_type: str*) → str
Get attachment format identifier for format and message combination.

Parameters **message_type** (*str*) – Message type for which to return the format identifier

Returns Issue credential attachment format identifier

Return type str

property profile: *aries_cloudagent.core.profile.Profile*
Accessor for the current profile instance.

Returns The profile instance for this presentation exchange format

abstract async receive_pres(*message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres.V20Pres, *pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord)
Receive a presentation, from message in context on manager creation.

abstract classmethod validate_fields(*message_type: str*, *attachment_data: dict*) → None
Validate attachment data for specific message type and format.

abstract async verify_pres(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord) → *aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord*
Verify a presentation.

exception aries_cloudagent.protocols.present_proof.v2_0.formats.handler.**V20PresFormatHandlerError**(**args*, *error_co* *Optional[* = *None*, ***kwa*

Bases: *aries_cloudagent.core.error.BaseError*

Presentation exchange format error under present-proof protocol v2.0.

aries_cloudagent.protocols.present_proof.v2_0.handlers package

Submodules

aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_ack_handler module

Presentation ack message handler.

class aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_ack_handler.**V20PresAckHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for presentation acks.

async 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.v2_0.handlers.pres_handler module

Presentation message handler.

class aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_handler.**V20PresHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for presentations.

async **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.v2_0.handlers.pres_problem_report_handler module

Presentation problem report message handler.

class aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_problem_report_handler.**V20PresProblemReportHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for problem reports.

async **handle**(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Message handler logic for problem reports.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_proposal_handler module

Presentation proposal message handler.

class aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_proposal_handler.**V20PresProposalHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for presentation proposals.

async **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.v2_0.handlers.pres_request_handler module

Presentation request message handler.

class aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_request_handler.**V20PresRequestHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Message handler class for v2.0 presentation requests.

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

Message handler logic for v2.0 presentation requests.

Parameters

- **context** – request context
- **responder** – responder callback

aries_cloudagent.protocols.present_proof.v2_0.messages package

Submodules

aries_cloudagent.protocols.present_proof.v2_0.messages.pres module

A (proof) presentation content message.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres.**V20Pres**(*_id: Optional[str] = None*, **, comment: Optional[str] = None*, *formats: Optional[Sequence[*aries_cloudagent.prot *= None*, *presentations_attach: Optional[Sequence[*aries_cloudagent.mes *= None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a presentation.

class Meta

Bases: object

Presentation metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_handler.V20PresHandler'

message_type = 'present-proof/2.0/presentation'

schema_class = 'V20PresSchema'

attachment(*fmt: Optional[aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat.Format] = None*) → dict

Return attached presentation item.

Parameters **fmt** – format of attachment in list to decode and return

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres.**V20PresSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Presentation schema.

class Meta

Bases: object

Presentation schema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.v2_0.messages.pres.V20Pres*

comment

formats

presentations_attach

validate_fields(*data*, ***kwargs*)

Validate presentation attachment per format.

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_ack module

Represents an explicit RFC 15 ack message, adopted into present-proof protocol.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_ack.**V20PresAck**(*status: Optional[str] = None*, *verification_result: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.protocols.notification.v1_0.messages.ack.V10Ack*

Base class representing an explicit ack message for present-proof protocol.

class Meta

Bases: object

V20PresAck metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_ack_handler.V20PresAckHandler'

message_type = 'present-proof/2.0/ack'

schema_class = 'V20PresAckSchema'

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_ack.**V20PresAckSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for V20PresAck class.

class **Meta**

Bases: object

V20PresAck schema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.v2_0.messages.pres_ack.V20PresAck*

verification_result

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format module

Credential format inner object.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.**FormatSpec**(*aries*, *handler*)

Bases: tuple

property **aries**

Alias for field number 0

property **handler**

Alias for field number 1

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.**V20PresFormat**(***, *attach_id: Optional[str] = None*, *format_: Optional[str] = None*)

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

Present-proof protocol message attachment format.

class **Format**(*value*)

Bases: enum.Enum

Attachment format.

DIF = FormatSpec(aries='dif/',
handler=<aries_cloudagent.utils.classloader.DeferLoad object>)

INDY = FormatSpec(aries='hlindy/', handler=<aries_cloudagent.utils.classloader.DeferLoad object>)

property api: str
Admin API specifier.

property aries: str
Accessor for aries identifier.

classmethod get(*label: Union[str, aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat.Format]*)
Get format enum for label.

get_attachment_data(*formats: Sequence[aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat]*, *attachments: Sequence[aries_cloudagent.messaging.decorators.attach_decorator.AttachDecorator]*)
Find attachment of current format, decode and return its content.

property handler: Type[V20PresFormatHandler]
Accessor for presentation exchange format handler.

validate_fields(*message_type: str*, *attachment_data: Mapping*)
Raise ValidationError for invalid attachment formats.

class Meta
Bases: object

Present-proof protocol message attachment format metadata.

schema_class = 'V20PresFormatSchema'

property format: str
Return format.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.**V20PresFormatSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Present-proof protocol message attachment format schema.

class Meta
Bases: object

Present-proof protocol message attachment format schema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat*

attach_id

format_

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_problem_report module

A problem report message.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_problem_report.**ProblemReportReason**(*val*
Bases: enum.Enum

Supported reason codes.

ABANDONED = 'abandoned'

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_problem_report.**V20PresProblemReport**(*

Bases: *aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReport*

Class representing a problem report message.

class Meta
Bases: object

Problem report metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_problem_report_handler.V20PresProblemReportHandler'

message_type = 'present-proof/2.0/problem-report'

schema_class = 'V20PresProblemReportSchema'

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_problem_report.**V20PresProblemReportSc**

Bases: marshmallow.

Problem report schema.

class Meta
Bases: object

Schema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.v2_0.messages.pres_problem_report.V20PresProblemReport*

validate_fields(*data*, ***kwargs*)
Validate schema fields.

Parameters **data** – The data to validate

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal module

A presentation proposal content message.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal.**V20PresProposal**(*_id: Optional[str] = None*, *, *comment: Optional[str] = None*, *formats: Optional[Sequence = None*, *proposals_attach: Optional[Sequence = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a presentation proposal.

class Meta

Bases: object

V20PresProposal metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_proposal_handler.V20PresProposalHandler'

message_type = 'present-proof/2.0/propose-presentation'

schema_class = 'V20PresProposalSchema'

attachment(*fmt: Optional[*aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat.Format*] = None*) → dict

Return attached proposal item.

Parameters **fmt** – format of attachment in list to decode and return

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal.**V20PresProposalSchema**(**args: Any*, ***kwarg Any*)

Bases: marshmallow.

Presentation proposal schema.

class Meta

Bases: object

Presentation proposal schema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal.V20PresProposal*

comment

formats

proposals_attach

validate_fields(*data*, ***kwargs*)
Validate proposal attachment per format.

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_request module

A presentation request content message.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_request.**V20PresRequest**(*_id: Optional[str] = None*, ***, *comment: Optional[str] = None*, *will_confirm: Optional[bool] = None*, *formats: Optional[Sequence[a = None*, *request_presentation Optional[Sequence[a = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a presentation request.

class Meta
Bases: object

V20PresRequest metadata.

handler_class = 'aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_request_handler.V20PresRequestHandler'

message_type = 'present-proof/2.0/request-presentation'

schema_class = 'V20PresRequestSchema'

attachment(*fmt: Optional[aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format.V20PresFormat.Format] = None*) → dict

Return attached presentation request item.

Parameters **fmt** – format of attachment in list to decode and return

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_request.**V20PresRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Presentation request schema.

class Meta

Bases: object

V20PresRequest schema metadata.

model_class

alias of *aries_cloudagent.protocols.present_proof.v2_0.messages.pres_request.V20PresRequest*

comment

formats

request_presentations_attach

validate_fields(*data*, ***kwargs*)

Validate proposal attachment per format.

will_confirm

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_webhook module

v2.0 Presentation exchange record webhook.

class aries_cloudagent.protocols.present_proof.v2_0.messages.pres_webhook.**V20PresExRecordWebhook**(***kwarg*

Bases: object

Class representing a state only Presentation exchange record webhook.

aries_cloudagent.protocols.present_proof.v2_0.models package

Package-wide data and code.

Submodules

aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange module

Presentation exchange record.

class aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.**V20PresExRecord**(*, *pres_ex_id: Optional[str] = None*, *connection_id: Optional[str] = None*, *thread_id: Optional[str] = None*, *initiator: Optional[str] = None*, *role: Optional[str] = None*, *state: Optional[str] = None*, *pres_proposal: Optional[Union[aries Mapping]] = None*, *pres_request: Optional[Union[aries Mapping]] = None*, *pres: Optional[Union[aries Mapping]] = None*, *verified: Op-*

Represents a v2.0 presentation exchange.

INITIATOR_EXTERNAL = 'external'

INITIATOR_SELF = 'self'

class Meta
Bases: object

V20PresExRecord metadata.

schema_class = 'V20PresExRecordSchema'

RECORD_ID_NAME = 'pres_ex_id'

RECORD_TOPIC: Optional[str] = 'present_proof_v2_0'

RECORD_TYPE = 'pres_ex_v20'

ROLE_PROVER = 'prover'

ROLE_VERIFIER = 'verifier'

STATE_ABANDONED = 'abandoned'

STATE_DONE = 'done'

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'

TAG_NAMES = {'thread_id'}

property by_format: Mapping
Record proposal, request, and presentation attachments by format.

async emit_event(*session:* aries_cloudagent.core.profile.ProfileSession, *payload: Optional[Any] = None*)
Emit an event.

Parameters

- **session** – The profile session to use
- **payload** – The event payload

property pres: *aries_cloudagent.protocols.present_proof.v2_0.messages.pres.V20Pres*
Accessor; get deserialized view.

property pres_ex_id: str
Accessor for the ID associated with this exchange record.

property pres_proposal:
aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal.V20PresProposal
Accessor; get deserialized view.

property pres_request:
aries_cloudagent.protocols.present_proof.v2_0.messages.pres_request.V20PresRequest
Accessor; get deserialized view.

property record_value: Mapping
Accessor for the JSON record value generated for this credential exchange.

async save_error_state(*session:* aries_cloudagent.core.profile.ProfileSession, *, *state: Optional[str] = None*, *reason: Optional[str] = None*, *log_params: Optional[Mapping[str, Any]] = None*, *log_override: bool = False*)
Save record error state if need be; log and swallow any storage error.

Parameters

- **session** – The profile session to use
- **reason** – A reason to add to the log
- **log_params** – Additional parameters to log
- **override** – Override configured logging regimen, print to stderr instead

class aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.**V20PresExRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for de/serialization of v2.0 presentation exchange records.

class Meta
Bases: object

V20PresExRecordSchema metadata.

model_class
alias of *aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord*

auto_present

auto_verify

by_format

connection_id

error_msg

initiator

pres

pres_ex_id

pres_proposal

pres_request

role

state

thread_id

verified

verified_msgs

Submodules

aries_cloudagent.protocols.present_proof.v2_0.controller module

Protocol controller for present proof v2_0.

class aries_cloudagent.protocols.present_proof.v2_0.controller.**Controller**(*protocol: str*)

Bases: object

Present proof v2_0 protocol controller.

determine_goal_codes() → Sequence[str]

Return defined goal_codes.

aries_cloudagent.protocols.present_proof.v2_0.manager module

Classes to manage presentations.

class aries_cloudagent.protocols.present_proof.v2_0.manager.**V20PresManager**(*profile:* aries_cloudagent.core.profile.Profile)

Bases: object

Class for managing presentations.

async create_bound_request(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord, *request_data: Optional[dict] = None*, *comment: Optional[str] = None*)

Create a presentation request bound to a proposal.

Parameters

- **pres_ex_record** – Presentation exchange record for which to create presentation request
- **comment** – Optional human-readable comment pertaining to request creation

Returns A tuple (updated presentation exchange record, presentation request message)

async create_exchange_for_proposal(*connection_id: str*, *pres_proposal_message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal.V20PresPr
auto_present: Optional[bool] = None)

Create a presentation exchange record for input presentation proposal.

Parameters

- **connection_id** – connection identifier
- **pres_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

async create_exchange_for_request(*connection_id: str*, *pres_request_message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres_request.V20PresRequ
auto_verify: Optional[bool] = None)

Create a presentation exchange record for input presentation request.

Parameters

- **connection_id** – connection identifier

- **pres_request_message** – presentation request to use in creating exchange record, extracting indy proof request and thread id

Returns Presentation exchange record, updated

async create_pres(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord, *request_data: dict = {}*, **, comment: Optional[str] = None*) → Tuple[*aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord*, *aries_cloudagent.protocols.present_proof.v2_0.messages.pres.V20Pres*]

Create a presentation.

Parameters

- **pres_ex_record** – record to update
- **requested_credentials** – indy formatted requested_credentials
- **comment** – optional human-readable comment
- **format** – presentation format

Example *requested_credentials* format, mapping proof request referents (uuid) to wallet referents (cred id):

```
{
    "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)

async receive_pres(*message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres.V20Pres, *connection_record: Optional[*aries_cloudagent.connections.models.conn_record.ConnRecord*]*, *oob_record: Optional[*aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record.OobRecord*]*)

Receive a presentation, from message in context on manager creation.

Returns presentation exchange record, retrieved and updated

async receive_pres_ack(*message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres_ack.V20PresAck, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord)

Receive a presentation ack, from message in context on manager creation.

Returns presentation exchange record, retrieved and updated

async receive_pres_proposal(*message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal.V20PresProposal, *conn_record:* aries_cloudagent.connections.models.conn_record.ConnRecord)

Receive a presentation proposal from message in context on manager creation.

Returns Presentation exchange record, created

async receive_pres_request(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord)

Receive a presentation request.

Parameters **pres_ex_record** – presentation exchange record with request to receive

Returns The presentation exchange record, updated

async receive_problem_report(*message:* aries_cloudagent.protocols.present_proof.v2_0.messages.pres_problem_report.V20PresPro *connection_id: str*)

Receive problem report.

Returns presentation exchange record, retrieved and updated

async send_pres_ack(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord, *responder: Optional[*aries_cloudagent.messaging.responder.BaseResponder*] = None*)

Send acknowledgement of presentation receipt.

Parameters **pres_ex_record** – presentation exchange record with thread id

async verify_pres(*pres_ex_record:* aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange.V20PresExRecord, *responder: Optional[*aries_cloudagent.messaging.responder.BaseResponder*] = None*)

Verify a presentation.

Parameters **pres_ex_record** – presentation exchange record with presentation request and presentation to verify

Returns presentation exchange record, updated

exception aries_cloudagent.protocols.present_proof.v2_0.manager.**V20PresManagerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Presentation error.

aries_cloudagent.protocols.present_proof.v2_0.message_types module

Message and inner object type identifiers for present-proof protocol v2.0.

aries_cloudagent.protocols.present_proof.v2_0.routes module

Admin routes for presentations.

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20CredentialsFetchQueryStringSchema**(**args: Any*, ***kwargs Any*)

Bases: marshmallow.

Parameters and validators for credentials fetch request query string.

count

extra_query

referent

start

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresCreateRequestRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for creating a proof request free of any connection.

auto_verify

comment

presentation_request

trace

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresExIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters for request taking presentation exchange id.

pres_ex_id

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresExRecordListQueryStringSchema**(**args: Any*, ***kwargs Any*)

Bases: marshmallow.

Parameters and validators for presentation exchange list query.

connection_id

role

state

thread_id

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresExRecordListSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Result schema for a presentation exchange query.

results

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresProblemReportRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending problem report.

description

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresProposalByFormatSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema for presentation proposal per format.

dif

indy

validate_fields(*data*, ***kwargs*)

Validate schema fields: data must have at least one format.

Parameters **data** – The data to validate

Raises **ValidationError** – if data has no formats

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresProposalRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending a presentation proposal admin message.

auto_present

comment

connection_id

presentation_proposal

trace

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresRequestByFormatSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Presentation request per format.

dif

indy

validate_fields(*data*, ***kwargs*)
Validate schema fields: data must have at least one format.

Parameters **data** – The data to validate

Raises **ValidationError** – if data has no formats

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresSendRequestRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for sending a proof request on a connection.

connection_id

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresSpecByFormatRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Presentation specification schema by format, for send-presentation request.

dif

indy

validate_fields(*data*, ***kwargs*)
Validate schema fields: specify exactly one format.

Parameters **data** – The data to validate

Raises **ValidationError** – if data does not have exactly one format.

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresentProofModuleResponseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Response schema for Present Proof Module.

class aries_cloudagent.protocols.present_proof.v2_0.routes.**V20PresentationSendRequestToProposalSchema**(**

Bases: marshmallow.

Request schema for sending a proof request bound to a proposal.

auto_verify

trace

aries_cloudagent.protocols.present_proof.v2_0.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.present_proof.v2_0.routes.**process_vcrecords_return_list**(*vc_records: Sequence[aries_clou record_ids: set*) → Tuple[Sequence[*aries* set]

Return list of non-duplicate VCRecords.

async aries_cloudagent.protocols.present_proof.v2_0.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

async aries_cloudagent.protocols.present_proof.v2_0.routes.**retrieve_uri_list_from_schema_filter**(*schema_u Sequence[S* → Sequence[st

Retrieve list of schema uri from uri_group.

Submodules

aries_cloudagent.protocols.present_proof.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.problem_report package

Subpackages

aries_cloudagent.protocols.problem_report.v1_0 package

async aries_cloudagent.protocols.problem_report.v1_0.**internal_error**(*err: aries_cloudagent.core.error.BaseError, http_error_class, record: Union[aries_cloudagent.connections.models.c aries_cloudagent.messaging.models.base_reco outbound_handler: Coroutine, code: Optional[str] = None*)

Send problem report and raise corresponding HTTP error.

Submodules

aries_cloudagent.protocols.problem_report.v1_0.handler module

Generic problem report handler.

class aries_cloudagent.protocols.problem_report.v1_0.handler.**ProblemReportHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Problem report handler class.

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

Handle problem report message.

Parameters

- **context** – Request context
- **responder** – Responder used to reply

aries_cloudagent.protocols.problem_report.v1_0.message module

Represents a generic problem report message.

class aries_cloudagent.protocols.problem_report.v1_0.message.**ProblemReport**(*, *description: Optional[Mapping[str, str]] = None*, *problem_items: Optional[Sequence[Mapping[str, str]]] = None*, *who_retries: Optional[str] = None*, *fix_hint: Optional[Mapping[str, str]] = None*, *impact: Optional[str] = None*, *where: Optional[str] = None*, *noticed_time: Optional[str] = None*, *tracking_uri: Optional[str] = None*, *escalation_uri: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Base class representing a generic problem report message.

class Meta
Bases: object

Problem report metadata.

handler_class = 'aries_cloudagent.protocols.problem_report.v1_0.handler.ProblemReportHandler'

message_type = 'notification/1.0/problem-report'

schema_class = 'ProblemReportSchema'

class aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReportSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Schema for ProblemReport base class.

class Meta
Bases: object

Problem report schema metadata.

model_class
alias of aries_cloudagent.protocols.problem_report.v1_0.message.ProblemReport

description

escalation_uri

fix_hint

impact

problem_items

time_noticed

tracking_uri

validate_fields(*data*, ***kwargs*)
Validate schema fields.

Parameters **data** – The data to validate

Raises **ValidationError** – if data has neither indy nor ld_proof

where

who_retries

aries_cloudagent.protocols.problem_report.v1_0.message_types module

Message type identifiers for problem reports.

Submodules

aries_cloudagent.protocols.problem_report.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.revocation_notification package

Subpackages

aries_cloudagent.protocols.revocation_notification.v1_0 package

Subpackages

aries_cloudagent.protocols.revocation_notification.v1_0.handlers package

Submodules

aries_cloudagent.protocols.revocation_notification.v1_0.handlers.revoke_handler module

Handler for revoke message.

class aries_cloudagent.protocols.revocation_notification.v1_0.handlers.revoke_handler.**RevokeHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for revoke message.

RECIEVED_TOPIC = 'acapy::revocation-notification::received'

WEBHOOK_TOPIC = 'acapy::webhook::revocation-notification'

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

Handle revoke message.

aries_cloudagent.protocols.revocation_notification.v1_0.messages package

Submodules

aries_cloudagent.protocols.revocation_notification.v1_0.messages.revoke module

Revoke message.

class aries_cloudagent.protocols.revocation_notification.v1_0.messages.revoke.**Revoke**(*, *thread_id: str*, *comment: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing revoke message.

class Meta

Bases: object

Revoke Meta.

handler_class = 'aries_cloudagent.protocols.revocation_notification.v1_0.handlers.revoke_handler.RevokeHandler'

message_type = 'revocation_notification/1.0/revoke'

schema_class = 'RevokeSchema'

class aries_cloudagent.protocols.revocation_notification.v1_0.messages.revoke.**RevokeSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema of Revoke message.

class Meta

Bases: object

RevokeSchema Meta.

model_class

alias of *aries_cloudagent.protocols.revocation_notification.v1_0.messages.revoke.Revoke*

comment

thread_id

aries_cloudagent.protocols.revocation_notification.v1_0.models package

Submodules

aries_cloudagent.protocols.revocation_notification.v1_0.models.rev_notification_record module

Store revocation notification details until revocation is published.

class aries_cloudagent.protocols.revocation_notification.v1_0.models.rev_notification_record.RevNotifica

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

Revocation Notification Record.

class Meta
Bases: object

RevNotificationRecord Meta.

schema_class = 'RevNotificationRecordSchema'

RECORD_ID_NAME = 'revocation_notification_id'

RECORD_TYPE = 'revocation_notification'

TAG_NAMES = {'connection_id', 'cred_rev_id', 'rev_reg_id', 'version'}

async classmethod query_by_ids(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_rev_id: str*, *rev_reg_id: str*) → *aries_cloudagent.protocols.revocation_notification.v1_0.models.rev_notification_record.*

Retrieve revocation notification record by cred rev id and/or rev reg id.

Parameters

- **session** – the profile session to use
- **cred_rev_id** – the cred rev id by which to filter
- **rev_reg_id** – the rev reg id by which to filter

async classmethod query_by_rev_reg_id(*session:* aries_cloudagent.core.profile.ProfileSession, *rev_reg_id: str*) → Sequence[*aries_cloudagent.protocols.revocation_notification.v1_0.models.rev_no*

Retrieve revocation notification records by rev reg id.

Parameters

- **session** – the profile session to use
- **rev_reg_id** – the rev reg id by which to filter

property record_value: dict

Return record value.

property revocation_notification_id: Optional[str]

Return record id.

to_message()

Return a revocation notification constructed from this record.

class aries_cloudagent.protocols.revocation_notification.v1_0.models.rev_notification_record.RevNotifica

Bases: marshmallow.

Revocation Notification Record Schema.

class Meta

Bases: object

RevNotificationRecordSchema Meta.

model_class = 'RevNotificationRecord'

comment

connection_id

cred_rev_id

rev_reg_id

thread_id

version

Submodules

aries_cloudagent.protocols.revocation_notification.v1_0.message_types module

Message type identifiers for Revocation Notification protocol.

aries_cloudagent.protocols.revocation_notification.v1_0.routes module

Routes for revocation notification.

async aries_cloudagent.protocols.revocation_notification.v1_0.routes.**on_pending_cleared**(*profile:* aries_cloudagent.co *event:* aries_cloudagent.co

Handle pending cleared event.

async aries_cloudagent.protocols.revocation_notification.v1_0.routes.**on_revocation_published**(*profile:* aries_cloudag *event:* aries_cloudag

Handle issuer revoke event.

aries_cloudagent.protocols.revocation_notification.v1_0.routes.**register_events**(*event_bus:* aries_cloudagent.core.event_bus

Register to handle events.

aries_cloudagent.protocols.revocation_notification.v2_0 package

Subpackages

aries_cloudagent.protocols.revocation_notification.v2_0.handlers package

Submodules

aries_cloudagent.protocols.revocation_notification.v2_0.handlers.revoke_handler module

Handler for revoke message.

class aries_cloudagent.protocols.revocation_notification.v2_0.handlers.revoke_handler.**RevokeHandler**

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for revoke message.

RECIEVED_TOPIC = 'acapy::revocation-notification-v2::received'

WEBHOOK_TOPIC = 'acapy::webhook::revocation-notification-v2'

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

Handle revoke message.

aries_cloudagent.protocols.revocation_notification.v2_0.messages package

Submodules

aries_cloudagent.protocols.revocation_notification.v2_0.messages.revoke module

Revoke message.

class aries_cloudagent.protocols.revocation_notification.v2_0.messages.revoke.**Revoke**(*, *revocation_format: str*, *credential_id: str*, *please_ack: Optional[*aries_cloudagent*] = None*, *comment: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing revoke message.

class **Meta**

Bases: object

Revoke Meta.

handler_class = 'aries_cloudagent.protocols.revocation_notification.v2_0.handlers.revoke_handler.RevokeHandler'

message_type = 'revocation_notification/2.0/revoke'

schema_class = 'RevokeSchema'

class aries_cloudagent.protocols.revocation_notification.v2_0.messages.revoke.**RevokeSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema of Revoke message.

class **Meta**

Bases: object

RevokeSchema Meta.

model_class
alias of *aries_cloudagent.protocols.revocation_notification.v2_0.messages.revoke.Revoke*

comment

credential_id

please_ack

revocation_format

aries_cloudagent.protocols.revocation_notification.v2_0.models package

Submodules

aries_cloudagent.protocols.revocation_notification.v2_0.models.rev_notification_record module

Store revocation notification details until revocation is published.

class aries_cloudagent.protocols.revocation_notification.v2_0.models.rev_notification_record.RevNotifica

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

Revocation Notification Record.

class Meta

Bases: object

RevNotificationRecord Meta.

schema_class = 'RevNotificationRecordSchema'

RECORD_ID_NAME = 'revocation_notification_id'

RECORD_TYPE = 'revocation_notification'

TAG_NAMES = {'connection_id', 'cred_rev_id', 'rev_reg_id', 'version'}

async classmethod query_by_ids(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_rev_id: str*, *rev_reg_id: str*) → *aries_cloudagent.protocols.revocation_notification.v2_0.models.rev_notification_record.*

Retrieve revocation notification record by cred rev id and/or rev reg id.

Parameters

- **session** – the profile session to use
- **cred_rev_id** – the cred rev id by which to filter
- **rev_reg_id** – the rev reg id by which to filter

async classmethod query_by_rev_reg_id(*session:* aries_cloudagent.core.profile.ProfileSession, *rev_reg_id: str*) → Sequence[*aries_cloudagent.protocols.revocation_notification.v2_0.models.rev_no*

Retrieve revocation notification records by rev reg id.

Parameters

- **session** – the profile session to use
- **rev_reg_id** – the rev reg id by which to filter

property record_value: dict

Return record value.

property revocation_notification_id: Optional[str]

Return record id.

to_message()

Return a revocation notification constructed from this record.

class aries_cloudagent.protocols.revocation_notification.v2_0.models.rev_notification_record.RevNotifica

Bases: marshmallow.

Revocation Notification Record Schema.

class Meta

Bases: object

RevNotificationRecordSchema Meta.

model_class = 'RevNotificationRecord'

comment

connection_id

cred_rev_id

rev_reg_id

thread_id

version

Submodules

aries_cloudagent.protocols.revocation_notification.v2_0.message_types module

Message type identifiers for Revocation Notification protocol.

aries_cloudagent.protocols.revocation_notification.v2_0.routes module

Routes for revocation notification.

async aries_cloudagent.protocols.revocation_notification.v2_0.routes.**on_pending_cleared**(*profile:* aries_cloudagent.co *event:* aries_cloudagent.co

Handle pending cleared event.

async aries_cloudagent.protocols.revocation_notification.v2_0.routes.**on_revocation_published**(*profile:* aries_cloudag *event:* aries_cloudag

Handle issuer revoke event.

aries_cloudagent.protocols.revocation_notification.v2_0.routes.**register_events**(*event_bus:* aries_cloudagent.core.event_bus

Register to handle events.

Submodules

aries_cloudagent.protocols.revocation_notification.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.routing package

Subpackages

aries_cloudagent.protocols.routing.v1_0 package

Subpackages

aries_cloudagent.protocols.routing.v1_0.handlers package

Submodules

aries_cloudagent.protocols.routing.v1_0.handlers.forward_handler module

aries_cloudagent.protocols.routing.v1_0.handlers.route_query_request_handler module

Handler for incoming route-query-request messages.

class aries_cloudagent.protocols.routing.v1_0.handlers.route_query_request_handler.RouteQueryRequestHandler

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming route-query-request messages.

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

Message handler implementation.

aries_cloudagent.protocols.routing.v1_0.handlers.route_query_response_handler module

Handler for incoming route-query-response messages.

class aries_cloudagent.protocols.routing.v1_0.handlers.route_query_response_handler.RouteQueryResponseHandler

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming route-query-response messages.

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

Message handler implementation.

aries_cloudagent.protocols.routing.v1_0.handlers.route_update_request_handler module

Handler for incoming route-update-request messages.

class aries_cloudagent.protocols.routing.v1_0.handlers.route_update_request_handler.RouteUpdateRequestHandler

Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Handler for incoming route-update-request messages.

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

Message handler implementation.

aries_cloudagent.protocols.routing.v1_0.handlers.route_update_response_handler module

aries_cloudagent.protocols.routing.v1_0.messages package

Submodules

aries_cloudagent.protocols.routing.v1_0.messages.forward module

Represents a forward message.

class aries_cloudagent.protocols.routing.v1_0.messages.forward.**Forward**(*, *to: Optional[str] = None*, *msg: Optional[Union[dict, str]] = None*, ***kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Represents a request to forward a message to a connected agent.

class Meta
Bases: object

Forward metadata.

handler_class = 'aries_cloudagent.protocols.routing.v1_0.handlers.forward_handler.ForwardHandler'

message_type = 'routing/1.0/forward'

schema_class = 'ForwardSchema'

class aries_cloudagent.protocols.routing.v1_0.messages.forward.**ForwardSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Forward message schema used in serialization/deserialization.

class Meta
Bases: object

ForwardSchema metadata.

model_class
alias of *aries_cloudagent.protocols.routing.v1_0.messages.forward.Forward*

handle_str_message(*data*, ***kwargs*)
Accept string value for msg, as produced by previous implementation.

msg

to

aries_cloudagent.protocols.routing.v1_0.messages.route_query_request module

Query existing forwarding routes.

class aries_cloudagent.protocols.routing.v1_0.messages.route_query_request.**RouteQueryRequest**(**, filter: Optional[dict] = None, paginate: Optional[aries_c = None, **kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Query existing routes from a routing agent.

class Meta
Bases: object

RouteQueryRequest metadata.

handler_class = 'aries_cloudagent.protocols.routing.v1_0.handlers.route_query_request_handler.RouteQueryRequestHandler'

message_type = 'routing/1.0/route-query-request'

schema_class = 'RouteQueryRequestSchema'

class aries_cloudagent.protocols.routing.v1_0.messages.route_query_request.**RouteQueryRequestSchema**(**args*, ***kwargs*)

Bases: marshmallow.

RouteQueryRequest message schema used in serialization/deserialization.

class Meta

Bases: object

RouteQueryRequestSchema metadata.

model_class

alias of *aries_cloudagent.protocols.routing.v1_0.messages.route_query_request.RouteQueryRequest*

filter

paginate

aries_cloudagent.protocols.routing.v1_0.messages.route_query_response module

Return existing forwarding routes in response to a query.

class aries_cloudagent.protocols.routing.v1_0.messages.route_query_response.**RouteQueryResponse**(**, routes: Optional[Seq = None, paginated: Optional[arie = None, **kwargs*)

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Return existing routes from a routing agent.

class Meta

Bases: object

RouteQueryResponse metadata.

handler_class = 'aries_cloudagent.protocols.routing.v1_0.handlers.route_query_response_handler.RouteQueryResponseHandler'

message_type = 'routing/1.0/route-query-response'

schema_class = 'RouteQueryResponseSchema'

class aries_cloudagent.protocols.routing.v1_0.messages.route_query_response.RouteQueryResponseSchema(*ar
An
**k
An

Bases: marshmallow.

RouteQueryResponse message schema used in serialization/deserialization.

class Meta
Bases: object

RouteQueryResponseSchema metadata.

model_class
alias of aries_cloudagent.protocols.routing.v1_0.messages.route_query_response.RouteQueryResponse

paginated

routes

aries_cloudagent.protocols.routing.v1_0.messages.route_update_request module

Request to update forwarding routes.

class aries_cloudagent.protocols.routing.v1_0.messages.route_update_request.RouteUpdateRequest(*, up-
dates:
Op-
tional[Seq
=
None,
**kwargs)

Bases: aries_cloudagent.messaging.agent_message.AgentMessage

Request to existing routes with a routing agent.

class Meta
Bases: object

RouteUpdateRequest metadata.

handler_class = 'aries_cloudagent.protocols.routing.v1_0.handlers.route_update_request_handler.RouteUpdateRequestHandler'

message_type = 'routing/1.0/route-update-request'

schema_class = 'RouteUpdateRequestSchema'

class aries_cloudagent.protocols.routing.v1_0.messages.route_update_request.RouteUpdateRequestSchema(*ar
An
**k
An

Bases: marshmallow.

RouteUpdateRequest message schema used in serialization/deserialization.

class Meta
Bases: object

RouteUpdateRequestSchema metadata.

model_class
alias of *aries_cloudagent.protocols.routing.v1_0.messages.route_update_request.RouteUpdateRequest*

updates

aries_cloudagent.protocols.routing.v1_0.messages.route_update_response module

Response for a route update request.

class aries_cloudagent.protocols.routing.v1_0.messages.route_update_response.**RouteUpdateResponse**(*, *updated: Optional[S* = *None*, ***kwarg*

Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Response for a route update request.

class Meta
Bases: object

RouteUpdateResponse metadata.

handler_class = 'aries_cloudagent.protocols.routing.v1_0.handlers.route_update_response_handler.RouteUpdateResponseHandler'

message_type = 'routing/1.0/route-update-response'

schema_class = 'RouteUpdateResponseSchema'

class aries_cloudagent.protocols.routing.v1_0.messages.route_update_response.**RouteUpdateResponseSchema**(

Bases: marshmallow.

RouteUpdateResponse message schema used in serialization/deserialization.

class Meta
Bases: object

RouteUpdateResponseSchema metadata.

model_class
alias of *aries_cloudagent.protocols.routing.v1_0.messages.route_update_response.RouteUpdateResponse*

updated

aries_cloudagent.protocols.routing.v1_0.models package

Submodules

aries_cloudagent.protocols.routing.v1_0.models.paginate module

An object for containing the request pagination information.

class aries_cloudagent.protocols.routing.v1_0.models.paginate.**Paginate**(*, *limit: Optional[int] = None*, *offset: Optional[int] = None*, ***kwargs*)

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

Class representing the pagination details of a request.

class Meta

Bases: object

Paginate metadata.

schema_class = 'PaginateSchema'

class aries_cloudagent.protocols.routing.v1_0.models.paginate.**PaginateSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Paginate schema.

class Meta

Bases: object

PaginateSchema metadata.

model_class

alias of *aries_cloudagent.protocols.routing.v1_0.models.paginate.Paginate*

limit

offset

aries_cloudagent.protocols.routing.v1_0.models.paginated module

An object for containing the response pagination information.

class aries_cloudagent.protocols.routing.v1_0.models.paginated.**Paginated**(*, *start: Optional[int] = None*, *end: Optional[int] = None*, *limit: Optional[int] = None*, *total: Optional[int] = None*, ***kwargs*)

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

Class representing the pagination details of a response.

class Meta
Bases: object

Paginated metadata.

schema_class = 'PaginatedSchema'

class aries_cloudagent.protocols.routing.v1_0.models.paginated.PaginatedSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Paginated schema.

class Meta
Bases: object

PaginatedSchema metadata.

model_class
alias of aries_cloudagent.protocols.routing.v1_0.models.paginated.Paginated

end

limit

start

total

aries_cloudagent.protocols.routing.v1_0.models.route_query_result module

An object for containing returned route information.

class aries_cloudagent.protocols.routing.v1_0.models.route_query_result.RouteQueryResult(*, recipient_key: Optional[str] = None, **kwargs)

Bases: aries_cloudagent.messaging.models.base.BaseModel

Class representing route information returned by a route query.

class Meta
Bases: object

RouteQueryResult metadata.

schema_class = 'RouteQueryResultSchema'

class aries_cloudagent.protocols.routing.v1_0.models.route_query_result.RouteQueryResultSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

RouteQueryResult schema.

class Meta
Bases: object

RouteQueryResultSchema metadata.

model_class
alias of aries_cloudagent.protocols.routing.v1_0.models.route_query_result.RouteQueryResult

recipient_key

aries_cloudagent.protocols.routing.v1_0.models.route_record module

An object for containing information on an individual route.

class aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord(*, record_id: Optional[str] = None, role: Optional[str] = None, connection_id: Optional[str] = None, wallet_id: Optional[str] = None, recipient_key: Optional[str] = None, **kwargs)

Bases: aries_cloudagent.messaging.models.base_record.BaseRecord

Class representing stored route information.

class Meta
Bases: object

RouteRecord metadata.

schema_class = 'RouteRecordSchema'

RECORD_ID_NAME = 'record_id'

RECORD_TYPE = 'forward_route'

ROLE_CLIENT = 'client'

ROLE_SERVER = 'server'

TAG_NAMES = {'connection_id', 'recipient_key', 'role', 'wallet_id'}

property record_id: str
Get record ID.

property record_value: dict
Accessor for JSON record value.

async classmethod retrieve_by_connection_id(session: aries_cloudagent.core.profile.ProfileSession, connection_id: str) → aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRec

Retrieve a route record by connection ID.

Parameters

- **session** (*ProfileSession*) – session
- **connection_id** (*str*) – ID to look up

Returns retrieved route record

Return type *RouteRecord*

async classmethod retrieve_by_recipient_key(*session:* aries_cloudagent.core.profile.ProfileSession, *recipient_key: str*) → *aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRec*

Retrieve a route record by recipient key.

Parameters

- **session** (*ProfileSession*) – session
- **recipient_key** (*str*) – key to look up

Returns retrieved route record

Return type *RouteRecord*

class aries_cloudagent.protocols.routing.v1_0.models.route_record.**RouteRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

RouteRecord schema.

class Meta

Bases: object

RouteRecordSchema metadata.

model_class

alias of *aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord*

connection_id

recipient_key

record_id

role

validate_fields(*data*, ***kwargs*)

Validate schema fields.

Parameters **data** – The data to validate

Raises **ValidationError** – If any of the fields do not validate

wallet_id

aries_cloudagent.protocols.routing.v1_0.models.route_update module

An object for containing route information to be updated.

class aries_cloudagent.protocols.routing.v1_0.models.route_update.**RouteUpdate**(*, *recipient_key: Optional[str] = None*, *action: Optional[str] = None*, ***kwargs*)

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

Class representing a route update request.

ACTION_CREATE = 'create'

ACTION_DELETE = 'delete'

class Meta

Bases: object

RouteUpdate metadata.

schema_class = 'RouteUpdateSchema'

class aries_cloudagent.protocols.routing.v1_0.models.route_update.**RouteUpdateSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

RouteUpdate schema.

class Meta

Bases: object

RouteUpdateSchema metadata.

model_class

alias of *aries_cloudagent.protocols.routing.v1_0.models.route_update.RouteUpdate*

action

recipient_key

aries_cloudagent.protocols.routing.v1_0.models.route_updated module

An object for containing updated route information.

class aries_cloudagent.protocols.routing.v1_0.models.route_updated.**RouteUpdated**(*, *recipient_key: Optional[str] = None*, *action: Optional[str] = None*, *result: Optional[str] = None*, ***kwargs*)

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

Class representing a route update response.

class Meta

Bases: object

RouteUpdated metadata.

schema_class = 'RouteUpdatedSchema'

RESULT_CLIENT_ERROR = 'client_error'

RESULT_NO_CHANGE = 'no_change'

RESULT_SERVER_ERROR = 'server_error'

RESULT_SUCCESS = 'success'

class aries_cloudagent.protocols.routing.v1_0.models.route_updated.**RouteUpdatedSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

RouteUpdated schema.

class Meta

Bases: object

RouteUpdatedSchema metadata.

model_class

alias of *aries_cloudagent.protocols.routing.v1_0.models.route_updated.RouteUpdated*

action

recipient_key

result

Submodules

aries_cloudagent.protocols.routing.v1_0.manager module

Routing manager classes for tracking and inspecting routing records.

exception aries_cloudagent.protocols.routing.v1_0.manager.**RouteNotFoundError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.protocols.routing.v1_0.manager.RoutingManagerError*

Requested route was not found.

class aries_cloudagent.protocols.routing.v1_0.manager.**RoutingManager**(*profile:* aries_cloudagent.core.profile.Profile)

Bases: object

Class for handling routing records.

RECORD_TYPE = 'forward_route'

async create_route_record(*client_connection_id: Optional[str] = None*, *recipient_key: Optional[str] = None*, *internal_wallet_id: Optional[str] = None*) → *aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord*

Create and store a new RouteRecord.

Parameters

- **client_connection_id** – The ID of the connection record
- **recipient_key** – The recipient verkey of the route
- **internal_wallet_id** – The ID of the wallet record. Used for internal routing

Returns The new routing record

async delete_route_record(*route:* aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord)

Remove an existing route record.

async get_recipient(*recip_verkey: str*) → *aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord*

Resolve the recipient for a verkey.

Parameters **recip_verkey** – The verkey (“to”) of the incoming Forward message

Returns The *RouteRecord* associated with this verkey

async get_routes(*client_connection_id: Optional[str] = None*, *tag_filter: Optional[dict] = None*) → Sequence[*aries_cloudagent.protocols.routing.v1_0.models.route_record.RouteRecord*]

Fetch all routes associated with the current connection.

Parameters

- **client_connection_id** – The ID of the connection record
- **tag_filter** – An optional dictionary of tag filters

Returns A sequence of route records found by the query

async send_create_route(*router_connection_id: str*, *recip_key: str*, *outbound_handler: Coroutine*)
Create and send a route update request.

Returns: the current routing state (request or done)

async update_routes(*client_connection_id: str*, *updates: Sequence[*aries_cloudagent.protocols.routing.v1_0.models.route_update.RouteUpdate*]*) → Sequence[*aries_cloudagent.protocols.routing.v1_0.models.route_updated.RouteUpdated*]
Update routes associated with the current connection.

Parameters

- **client_connection_id** – The ID of the connection record
- **updates** – The sequence of route updates (create/delete) to perform.

exception aries_cloudagent.protocols.routing.v1_0.manager.**RoutingManagerError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Generic routing error.

aries_cloudagent.protocols.routing.v1_0.message_types module

Message type identifiers for Routing.

Submodules

aries_cloudagent.protocols.routing.definition module

Version definitions for this protocol.

aries_cloudagent.protocols.trustping package

Subpackages

aries_cloudagent.protocols.trustping.v1_0 package

Subpackages

aries_cloudagent.protocols.trustping.v1_0.handlers package

Submodules

aries_cloudagent.protocols.trustping.v1_0.handlers.ping_handler module

Ping handler.

class aries_cloudagent.protocols.trustping.v1_0.handlers.ping_handler.**PingHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Ping handler class.

async **handle**(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Handle ping message.

Parameters

- **context** – Request context
- **responder** – Responder used to reply

aries_cloudagent.protocols.trustping.v1_0.handlers.ping_response_handler module

Ping response handler.

class aries_cloudagent.protocols.trustping.v1_0.handlers.ping_response_handler.**PingResponseHandler**
Bases: *aries_cloudagent.messaging.base_handler.BaseHandler*

Ping response handler class.

async **handle**(*context:* aries_cloudagent.messaging.request_context.RequestContext, *responder:* aries_cloudagent.messaging.responder.BaseResponder)
Handle ping response message.

Parameters

- **context** – Request context
- **responder** – Responder used to reply

aries_cloudagent.protocols.trustping.v1_0.messages package

Submodules

aries_cloudagent.protocols.trustping.v1_0.messages.ping module

Represents a trust ping message.

class aries_cloudagent.protocols.trustping.v1_0.messages.ping.**Ping**(*, *response_requested: bool = True*, *comment: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.messaging.agent_message.AgentMessage*

Class representing a trustping message.

class **Meta**
Bases: object

Ping metadata.

handler_class = 'aries_cloudagent.protocols.trustping.v1_0.handlers.ping_handler.PingHandler'

message_type = 'trust_ping/1.0/ping'

schema_class = 'PingSchema'

class aries_cloudagent.protocols.trustping.v1_0.messages.ping.PingSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

Schema for Ping class.

class Meta

Bases: object

PingSchema metadata.

model_class

alias of aries_cloudagent.protocols.trustping.v1_0.messages.ping.Ping

comment

response_requested

aries_cloudagent.protocols.trustping.v1_0.messages.ping_response module

Represents an response to a trust ping message.

class aries_cloudagent.protocols.trustping.v1_0.messages.ping_response.PingResponse(*, comment: Optional[str] = None, **kwargs)

Bases: aries_cloudagent.messaging.agent_message.AgentMessage

Class representing a ping response.

class Meta

Bases: object

PingResponse metadata.

handler_class = 'aries_cloudagent.protocols.trustping.v1_0.handlers.ping_response_handler.PingResponseHandler'

message_type = 'trust_ping/1.0/ping_response'

schema_class = 'PingResponseSchema'

class aries_cloudagent.protocols.trustping.v1_0.messages.ping_response.PingResponseSchema(*args: Any, **kwargs: Any)

Bases: marshmallow.

PingResponse schema.

class Meta

Bases: object

PingResponseSchema metadata.

model_class
alias of *aries_cloudagent.protocols.trustping.v1_0.messages.ping_response.PingResponse*

comment

Submodules

aries_cloudagent.protocols.trustping.v1_0.message_types module

Message type identifiers for Trust Pings.

aries_cloudagent.protocols.trustping.v1_0.routes module

Trust ping admin routes.

class aries_cloudagent.protocols.trustping.v1_0.routes.**PingConnIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Path parameters and validators for request taking connection id.

conn_id

class aries_cloudagent.protocols.trustping.v1_0.routes.**PingRequestResponseSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for performing a ping.

thread_id

class aries_cloudagent.protocols.trustping.v1_0.routes.**PingRequestSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Request schema for performing a ping.

comment

aries_cloudagent.protocols.trustping.v1_0.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.protocols.trustping.v1_0.routes.**register**(*app: aiohttp.web.Application*)
Register routes.

Submodules

aries_cloudagent.protocols.trustping.definition module

Version definitions for this protocol.

Submodules

aries_cloudagent.protocols.didcomm_prefix module

DIDComm prefix management.

class aries_cloudagent.protocols.didcomm_prefix.**DIDCommPrefix**(*value*)
Bases: enum.Enum

Enum for DIDComm Prefix, old or new style, per Aries RFC 384.

NEW = 'https://didcomm.org'

OLD = 'did:sov:BzCbsNYhMrjHiqZDTUASHg;spec'

qualify(*msg_type: Optional[str] = None*) → str
Qualify input message type with prefix and separator.

classmethod qualify_all(*messages: dict*) → dict
Apply all known prefixes to a dictionary of message types.

static qualify_current(*slug: Optional[str] = None*) → str
Qualify input slug with prefix currently in effect and separator.

static set(*settings: Mapping*)
Set current DIDComm prefix value in environment.

static unqualify(*qual: str*) → str
Strip prefix and separator from input, if present, and return result.

aries_cloudagent.protocols.didcomm_prefix.**qualify**(*msg_type: str*, *prefix: str*)
Qualify a message type with a prefix, if unqualified.

aries_cloudagent.resolver package

Interfaces and base classes for DID Resolution.

async aries_cloudagent.resolver.**setup**(*context:*
aries_cloudagent.config.injection_context.InjectionContext)
Set up default resolvers.

Subpackages

aries_cloudagent.resolver.default package

Resolvers included in ACA-Py by Default.

Submodules

aries_cloudagent.resolver.default.indy module

aries_cloudagent.resolver.default.key module

Key DID Resolver.

Resolution is performed using the IndyLedger class.

class aries_cloudagent.resolver.default.key.**KeyDIDResolver**

Bases: *aries_cloudagent.resolver.base.BaseDIDResolver*

Key DID Resolver.

async setup(*context:* aries_cloudagent.config.injection_context.InjectionContext)

Perform required setup for Key DID resolution.

property supported_did_regex: Pattern

Return supported_did_regex of Key DID Resolver.

aries_cloudagent.resolver.default.universal module

HTTP Universal DID Resolver.

class aries_cloudagent.resolver.default.universal.**UniversalResolver**(*, *endpoint: Optional[str] = None*, *supported_did_regex: Optional[Pattern] = None*, *bearer_token: Optional[str] = None*)

Bases: *aries_cloudagent.resolver.base.BaseDIDResolver*

Universal DID Resolver with HTTP bindings.

async setup(*context:* aries_cloudagent.config.injection_context.InjectionContext)

Perform setup, populate supported method list, configuration.

property supported_did_regex: Pattern

Return supported methods regex.

aries_cloudagent.resolver.default.web module

Web DID Resolver.

class aries_cloudagent.resolver.default.web.**WebDIDResolver**
Bases: *aries_cloudagent.resolver.base.BaseDIDResolver*

Web DID Resolver.

async setup(*context:* aries_cloudagent.config.injection_context.InjectionContext)
Perform required setup for Web DID resolution.

property supported_did_regex: Pattern
Return supported_did_regex of Web DID Resolver.

Submodules

aries_cloudagent.resolver.base module

Base Class for DID Resolvers.

class aries_cloudagent.resolver.base.**BaseDIDResolver**(*type_: Optional[*aries_cloudagent.resolver.base.ResolverType*] = None*)
Bases: abc.ABC

Base Class for DID Resolvers.

property native
Return if this resolver is native.

async resolve(*profile:* aries_cloudagent.core.profile.Profile, *did: Union[str, pydid.DID]*, *service_accept: Optional[Sequence[str]] = None*) → dict
Resolve a DID using this resolver.

abstract async setup(*context:* aries_cloudagent.config.injection_context.InjectionContext)
Do asynchronous resolver setup.

property supported_did_regex: Pattern
Supported DID regex for matching this resolver to DIDs it can resolve.

Override this property with a class var or similar to use regex matching on DIDs to determine if this resolver supports a given DID.

property supported_methods: Sequence[str]
Return supported methods.

DEPRECATED: Use supported_did_regex instead.

async supports(*profile:* aries_cloudagent.core.profile.Profile, *did: str*) → bool
Return if this resolver supports the given DID.

Override this method to determine if this resolver supports a DID based on information other than just a regular expression; i.e. check a value in storage, query a resolver connection record, etc.

exception aries_cloudagent.resolver.base.**DIDMethodNotSupported**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.resolver.base.ResolverError*

Raised when no resolver is registered for a given did method.

exception aries_cloudagent.resolver.base.**DIDNotFound**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.resolver.base.ResolverError*

Raised when DID is not found in verifiable data registry.

class aries_cloudagent.resolver.base.**ResolutionMetadata**(*resolver_type:* aries_cloudagent.resolver.base.ResolverType, *resolver: str*, *retrieved_time: str*, *duration: int*)

Bases: tuple

Resolution Metadata.

property duration
Alias for field number 3

property resolver
Alias for field number 1

property resolver_type
Alias for field number 0

property retrieved_time
Alias for field number 2

serialize() → dict
Return serialized resolution metadata.

class aries_cloudagent.resolver.base.**ResolutionResult**(*did_document: dict*, *metadata:* aries_cloudagent.resolver.base.ResolutionMetadata)

Bases: object

Resolution Class to pack the DID Doc and the resolution information.

serialize() → dict
Return serialized resolution result.

exception aries_cloudagent.resolver.base.**ResolverError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base class for resolver exceptions.

class aries_cloudagent.resolver.base.**ResolverType**(*value*)

Bases: enum.Enum

Resolver Type declarations.

NATIVE = 'native'

NON_NATIVE = 'non-native'

aries_cloudagent.resolver.did_resolver module

the did resolver.

responsible for keeping track of all resolvers. more importantly retrieving did's from different sources provided by the method type.

class aries_cloudagent.resolver.did_resolver.**DIDResolver**(*resolvers: Optional[List[*aries_cloudagent.resolver.base.BaseDIDResolver *= None*)

Bases: object

did resolver singleton.

async dereference(*profile:* aries_cloudagent.core.profile.Profile, *did_url: str*, *, *document: Optional[pydid.doc.doc.BaseDIDDocument] = None*) → pydid.Resource

Dereference a DID URL to its corresponding DID Doc object.

register_resolver(*resolver:* aries_cloudagent.resolver.base.BaseDIDResolver)

Register a new resolver.

async resolve(*profile:* aries_cloudagent.core.profile.Profile, *did: Union[str, pydid.DID]*, *service_accept: Optional[Sequence[str]] = None*) → dict

Resolve a DID.

async resolve_with_metadata(*profile:* aries_cloudagent.core.profile.Profile, *did: Union[str, pydid.DID]*) → *aries_cloudagent.resolver.base.ResolutionResult*

Resolve a DID and return the ResolutionResult.

aries_cloudagent.resolver.routes module

Resolve did document admin routes.

"/resolver/resolve/{did}": {

"get": {

"responses": {

"200": {

"schema": { "$ref": "#/definitions/DIDDoc"

}, "description": null

}

}, "parameters": [

{ "in": "path", "name": "did", "required": true, "type": "string", "pattern": "did:([a-z]+):((?:[a-zA-Z0-9._-]*:)*[a-zA-Z0-9._-]+)", "description": "decentralize identifier(DID)", "example": "did:ted:WgWxqztrNooG92RXvxSTWv"

}

],

"tags": ["resolver"], "summary": "Retrieve doc for requested did", "produces": ["application/json"]

}

}

class aries_cloudagent.resolver.routes.**DIDMatchInfoSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Path parameters and validators for request taking DID.

did

class aries_cloudagent.resolver.routes.**ResolutionResultSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Result schema for did document query.

did_document

metadata

class aries_cloudagent.resolver.routes.**W3cDID**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.validate.

Validate value against w3c DID.

EXAMPLE = 'did:ted:WgWxqztrNooG92RXvxSTWv'

aries_cloudagent.resolver.routes.**post_process_routes**(*app: aiohttp.web.Application*)
Amend swagger API.

async aries_cloudagent.resolver.routes.**register**(*app: aiohttp.web.Application*)
Register routes.

aries_cloudagent.revocation package

Subpackages

aries_cloudagent.revocation.models package

Submodules

aries_cloudagent.revocation.models.indy module

Indy utilities for revocation.

class aries_cloudagent.revocation.models.indy.**NonRevocationInterval**(*fro: Optional[int] = None*, *to: Optional[int] = None*, ***kwargs*)
Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Indy non-revocation interval.

class Meta
Bases: object

NonRevocationInterval metadata.

schema_class = 'NonRevocationIntervalSchema'

covers(*timestamp: Optional[int] = None*) → bool
Whether input timestamp (default now) lies within non-revocation interval.

Parameters **timestamp** – time of interest

Returns whether input time satisfies non-revocation interval

timestamp() → bool
Return a timestamp that the non-revocation interval covers.

class aries_cloudagent.revocation.models.indy.**NonRevocationIntervalSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema to allow serialization/deserialization of non-revocation intervals.

class Meta
Bases: object

NonRevocationIntervalSchema metadata.

model_class
alias of *aries_cloudagent.revocation.models.indy.NonRevocationInterval*

fro

to

aries_cloudagent.revocation.models.issuer_cred_rev_record module

Issuer credential revocation information.

class aries_cloudagent.revocation.models.issuer_cred_rev_record.**IssuerCredRevRecord**(*, *record_id: Optional[str] = None*, *state: Optional[str] = None*, *cred_ex_id: Optional[str] = None*, *rev_reg_id: Optional[str] = None*, *cred_rev_id: Optional[str] = None*, *cred_def_id: Optional[str] = None*, *cred_ex_version: Optional[str] = None*, ***kwargs*)

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

Represents credential revocation information to retain post-issue.

class Meta

Bases: object

IssuerCredRevRecord metadata.

schema_class = 'IssuerCredRevRecordSchema'

RECORD_ID_NAME = 'record_id'

RECORD_TOPIC: Optional[str] = 'issuer_cred_rev'

RECORD_TYPE = 'issuer_cred_rev'

STATE_ISSUED = 'issued'

STATE_REVOKED = 'revoked'

TAG_NAMES = {'cred_def_id', 'cred_ex_id', 'cred_ex_version', 'cred_rev_id', 'rev_reg_id', 'state'}

VERSION_1 = '1'

VERSION_2 = '2'

async classmethod query_by_ids(*session:* aries_cloudagent.core.profile.ProfileSession, *, *cred_def_id: Optional[str] = None*, *rev_reg_id: Optional[str] = None*, *state: Optional[str] = None*) → Sequence[*aries_cloudagent.revocation.models.issuer_cred_rev_record.IssuerCredRevReco*

Retrieve issuer cred rev records by cred def id and/or rev reg id.

Parameters

- **session** – the profile session to use
- **cred_def_id** – the cred def id by which to filter
- **rev_reg_id** – the rev reg id by which to filter
- **state** – a state value by which to filter

property record_id: str

Accessor for the ID associated with this exchange.

async classmethod retrieve_by_cred_ex_id(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_ex_id: str*) → *aries_cloudagent.revocation.models.issuer_cred_rev_record.IssuerCredRev*

Retrieve an issuer cred rev record by rev reg id and cred rev id.

async classmethod retrieve_by_ids(*session:* aries_cloudagent.core.profile.ProfileSession, *rev_reg_id: str*, *cred_rev_id: str*, *, *for_update: bool = False*) → *aries_cloudagent.revocation.models.issuer_cred_rev_record.IssuerCredRevRecord*

Retrieve an issuer cred rev record by rev reg id and cred rev id.

async set_state(*session:* aries_cloudagent.core.profile.ProfileSession, *state: Optional[str] = None*)

Change the issuer cred rev record state (default issued).

class aries_cloudagent.revocation.models.issuer_cred_rev_record.**IssuerCredRevRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema to allow de/serialization of credential revocation records.

class Meta

Bases: object

IssuerCredRevRecordSchema metadata.

model_class

alias of *aries_cloudagent.revocation.models.issuer_cred_rev_record.IssuerCredRevRecord*

cred_def_id

cred_ex_id

cred_ex_version

cred_rev_id

record_id

`rev_reg_id`

`state`

aries_cloudagent.revocation.models.issuer_rev_reg_record module

Issuer revocation registry storage handling.

class aries_cloudagent.revocation.models.issuer_rev_reg_record.**IssuerRevRegRecord**(*, *record_id: Optional[str] = None*, *state: Optional[str] = None*, *cred_def_id: Optional[str] = None*, *error_msg: Optional[str] = None*, *issuer_did: Optional[str] = None*, *max_cred_num: Optional[int] = None*, *revoc_def_type: Optional[str] = None*, *revoc_reg_id: Optional[str] = None*, *revoc_reg_def: Optional[Union[aries_cloudagent.revocation.models.revocation_registry.RevocationRegistry, Mapping]] = None*, *revoc_reg_entry: Optional[Union[aries_cloudagent.revocation.models.revocation_registry_entry.RevocationRegistryEntry, Mapping]] = None*, *tag: Optional[str] = None*, *tails_hash: Optional[str] = None*, *tails_local_path: Optional[str]*

Class for managing local issuing revocation registries.

LOG_STATE_FLAG = 'debug.revocation'

class Meta

Bases: object

IssuerRevRegRecord metadata.

schema_class = 'IssuerRevRegRecordSchema'

RECORD_ID_NAME = 'record_id'

RECORD_TOPIC: Optional[str] = 'revocation_registry'

RECORD_TYPE = 'issuer_rev_reg'

REVOC_DEF_TYPE_CL = 'CL_ACCUM'

STATE_ACTIVE = 'active'

STATE_FULL = 'full'

STATE_GENERATED = 'generated'

STATE_INIT = 'init'

STATE_POSTED = 'posted'

TAG_NAMES = {'cred_def_id', 'issuer_did', 'revoc_def_type', 'revoc_reg_id', 'state'}

async clear_pending(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_rev_ids: Optional[Sequence[str]] = None*) → None

Clear pending revocations and save any resulting record change.

Parameters

- **session** – The profile session to use
- **cred_rev_ids** – Credential revocation identifiers to clear; default all

async fix_ledger_entry(*profile:* aries_cloudagent.core.profile.Profile, *apply_ledger_update: bool*, *genesis_transactions: str*) → Tuple[dict, dict, dict]

Fix the ledger entry to match wallet-recorded credentials.

async generate_registry(*profile:* aries_cloudagent.core.profile.Profile)

Create the revocation registry definition and tails file.

get_registry() → *aries_cloudagent.revocation.models.revocation_registry.RevocationRegistry*

Create a *RevocationRegistry* instance from this record.

property has_local_tails_file: bool

Check if a local copy of the tails file is available.

async mark_pending(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_rev_id: str*) → None

Mark a credential revocation id as revoked pending publication to ledger.

Parameters

- **session** – The profile session to use
- **cred_rev_id** – The credential revocation identifier for credential to revoke

async classmethod query_by_cred_def_id(*session:* aries_cloudagent.core.profile.ProfileSession, *cred_def_id: str*, *state: Optional[str] = None*) → Sequence[*aries_cloudagent.revocation.models.issuer_rev_reg_record.IssuerRev*

Retrieve issuer revocation registry records by credential definition ID.

Parameters

- **session** – The profile session to use
- **cred_def_id** – The credential definition ID to filter by
- **state** – A state value to filter by

async classmethod query_by_pending(*session:* aries_cloudagent.core.profile.ProfileSession) → Sequence[*aries_cloudagent.revocation.models.issuer_rev_reg_record.IssuerRevRegR*

Retrieve issuer revocation records with revocations pending.

Parameters **session** – The profile session to use

property record_id: str

Accessor for the record ID.

property record_value: Mapping

Accessor for JSON value properties of this revocation registry record.

async classmethod retrieve_by_revoc_reg_id(*session:* aries_cloudagent.core.profile.ProfileSession, *revoc_reg_id: str*, *for_update: bool = False*) → *aries_cloudagent.revocation.models.issuer_rev_reg_record.IssuerRevReg*

Retrieve a revocation registry record by revocation registry ID.

Parameters

- **session** – The profile session to use
- **revoc_reg_id** – The revocation registry ID
- **for_update** – Retrieve for update

property revoc_reg_def: *aries_cloudagent.indy.models.revocation.IndyRevRegDef*

Accessor; get deserialized.

property revoc_reg_entry: *aries_cloudagent.indy.models.revocation.IndyRevRegEntry*

Accessor; get deserialized.

async send_def(*profile:* aries_cloudagent.core.profile.Profile, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict

Send the revocation registry definition to the ledger.

async send_entry(*profile:* aries_cloudagent.core.profile.Profile, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*) → dict

Send a registry entry to the ledger.

async set_state(*session:* aries_cloudagent.core.profile.ProfileSession, *state: Optional[str] = None*)

Change the registry state (default full).

async set_tails_file_public_uri(*profile:* aries_cloudagent.core.profile.Profile, *tails_file_uri: str*)

Update tails file's publicly accessible URI.

async upload_tails_file(*profile:* aries_cloudagent.core.profile.Profile)

Upload the local tails file to the tails server.

class aries_cloudagent.revocation.models.issuer_rev_reg_record.**IssuerRevRegRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema to allow serialization/deserialization of issuer rev reg records.

class Meta

Bases: object

IssuerRevRegRecordSchema metadata.

model_class

alias of *aries_cloudagent.revocation.models.issuer_rev_reg_record.IssuerRevRegRecord*

cred_def_id

error_msg

issuer_did

max_cred_num

pending_pub

record_id

revoc_def_type

revoc_reg_def

revoc_reg_entry

revoc_reg_id

state

tag

tails_hash

tails_local_path

tails_public_uri

aries_cloudagent.revocation.models.revocation_registry module

Classes for managing a revocation registry.

class aries_cloudagent.revocation.models.revocation_registry.**RevocationRegistry**(*registry_id: Optional[str] = None*, ***, *cred_def_id: Optional[str] = None*, *issuer_did: Optional[str] = None*, *max_creds: Optional[int] = None*, *reg_def_type: Optional[str] = None*, *tag: Optional[str] = None*, *tails_local_path: Optional[str] = None*, *tails_public_uri: Optional[str] = None*, *tails_hash: Optional[str] = None*, *reg_def: Optional[dict] = None*)

Bases: object

Manage a revocation registry and tails file.

MAX_SIZE = 32768

MIN_SIZE = 4

property cred_def_id: str

Accessor for the credential definition ID.

classmethod from_definition(*revoc_reg_def: dict*, *public_def: bool*) → *aries_cloudagent.revocation.models.revocation_registry.RevocationRegistry*

Initialize a revocation registry instance from a definition.

async get_or_fetch_local_tails_path()
Get the local tails path, retrieving from the remote if necessary.

get_receiving_tails_local_path()
Make the local path to the tails file we download from remote URI.

has_local_tails_file() → bool
Test if the tails file exists locally.

property issuer_did: str
Accessor for the issuer DID.

property max_creds: int
Accessor for the maximum number of issued credentials.

property reg_def: dict
Accessor for the revocation registry definition.

property reg_def_type: str
Accessor for the revocation registry type.

property registry_id: str
Accessor for the revocation registry ID.

async retrieve_tails()
Fetch the tails file from the public URI.

property tag: str
Accessor for the tag part of the revoc. reg. ID.

property tails_hash: str
Accessor for the tails file hash.

property tails_local_path: str
Accessor for the tails file local path.

property tails_public_uri: str
Accessor for the tails file public URI.

Submodules

aries_cloudagent.revocation.error module

Revocation error classes.

exception aries_cloudagent.revocation.error.**RevocationError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Base exception for revocation-related errors.

exception aries_cloudagent.revocation.error.**RevocationNotSupportedError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.revocation.error.RevocationError*

Attempted to perform revocation-related operation where inapplicable.

exception aries_cloudagent.revocation.error.**RevocationRegistryBadSizeError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.revocation.error.RevocationError*

Attempted to create registry with maximum credentials too large or too small.

aries_cloudagent.revocation.indy module

aries_cloudagent.revocation.manager module

aries_cloudagent.revocation.recover module

Recover a revocation registry.

aries_cloudagent.revocation.recover.**LOGGER = <Logger aries_cloudagent.revocation.recover (WARNING)>**

This module calculates a new ledger accumulator, based on the revocation status on the ledger vs revocations recorded in the wallet. The calculated transaction can be written to the ledger to get the ledger back in sync with the wallet. This function can be used if there were previous revocation errors (i.e. the credential revocation was successfully written to the wallet but the ledger write failed.)

exception aries_cloudagent.revocation.recover.**RevocRecoveryException**

Bases: Exception

Raise exception generating the recovery transaction.

async aries_cloudagent.revocation.recover.**fetch_txns**(*genesis_txns*, *registry_id*)

Fetch tails file and revocation registry information.

async aries_cloudagent.revocation.recover.**generate_ledger_rrrecovery_txn**(*genesis_txns*, *registry_id*, *set_revoked*)

Generate a new ledger accum entry, based on wallet vs ledger revocation state.

aries_cloudagent.revocation.routes module

aries_cloudagent.revocation.util module

Revocation utilities.

async aries_cloudagent.revocation.util.**notify_pending_cleared_event**(*profile:* aries_cloudagent.core.profile.Profile, *rev_reg_id: str*)

Send notification of credential revoked as issuer.

async aries_cloudagent.revocation.util.**notify_revocation_entry_endorsed_event**(*profile:* aries_cloudagent.core.profile.Pro *rev_reg_id: str*, *meta_data: dict*)

Send notification for a revocation registry entry endorsement event.

async aries_cloudagent.revocation.util.**notify_revocation_entry_event**(*profile: aries_cloudagent.core.profile.Profile*, *issuer_rev_id: str*, *meta_data: dict*)

Send notification for a revocation registry entry event.

async aries_cloudagent.revocation.util.**notify_revocation_published_event**(*profile: aries_cloudagent.core.profile.Profile*, *rev_reg_id: str*, *crids: Sequence[str]*)

Send notification of credential revoked as issuer.

async aries_cloudagent.revocation.util.**notify_revocation_reg_endorsed_event**(*profile: aries_cloudagent.core.profile.Profile*, *rev_reg_id: str*, *meta_data: dict*)

Send notification for a revocation registry endorsement event.

async aries_cloudagent.revocation.util.**notify_revocation_reg_init_event**(*profile: aries_cloudagent.core.profile.Profile*, *issuer_rev_id: str*, *create_pending_rev_reg: bool = False*, *endorser_connection_id: Optional[str] = None*)

Send notification for a revocation registry init event.

aries_cloudagent.storage package

Subpackages

aries_cloudagent.storage.vc_holder package

Submodules

aries_cloudagent.storage.vc_holder.askar module

aries_cloudagent.storage.vc_holder.base module

Abstract interfaces for VC holder implementations.

class aries_cloudagent.storage.vc_holder.base.**IterVCRecordSearch**(*search: aries_cloudagent.storage.vc_holder.base.VCRecor*, *page_size: Optional[int] = None*)

Bases: `object`

A generic record search async iterator.

class aries_cloudagent.storage.vc_holder.base.**VCHolder**

Bases: `abc.ABC`

Abstract base class for a verifiable credential holder.

abstract build_type_or_schema_query(*uri_list: Sequence[str]*) → dict
Build and return backend-specific type_or_schema_query.

Parameters **uri_list** – List of schema uri from input_descriptor

abstract async delete_credential(*cred:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord)
Remove a previously-stored VC record.

Raises **StorageNotFoundError** – If the record is not found

abstract async retrieve_credential_by_given_id(*given_id: str*) → *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*
Fetch a VC record by its given ID ('id' property).

Raises **StorageNotFoundError** – If the record is not found

abstract async retrieve_credential_by_id(*record_id: str*) → *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*
Fetch a VC record by its record ID.

Raises **StorageNotFoundError** – If the record is not found

abstract search_credentials(*contexts: Optional[Sequence[str]] = None*, *types: Optional[Sequence[str]] = None*, *schema_ids: Optional[Sequence[str]] = None*, *issuer_id: Optional[str] = None*, *subject_ids: Optional[Sequence[str]] = None*, *proof_types: Optional[Sequence[str]] = None*, *given_id: Optional[str] = None*, *tag_query: Optional[Mapping] = None*) → *aries_cloudagent.storage.vc_holder.base.VCRecordSearch*
Start a new VC record search.

Parameters

- **contexts** – An inclusive list of JSON-LD contexts to match
- **types** – An inclusive list of JSON-LD types to match
- **schema_ids** – An inclusive list of credential schema identifiers
- **issuer_id** – The ID of the credential issuer
- **subject_ids** – The IDs of any credential subjects all of which to match
- **proof_types** – The signature suite types used for the proof objects.
- **given_id** – The given id of the credential
- **tag_query** – A tag filter clause

abstract async store_credential(*cred:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord)
Add a new VC record to the store.

Parameters **cred** – The VCRecord instance to store

Raises **StorageDuplicateError** – If the record_id is not unique

class aries_cloudagent.storage.vc_holder.base.**VCRecordSearch**
Bases: abc.ABC

A VC record search in progress.

async close()
Dispose of the search query.

abstract async fetch(*max_count: Optional[int] = None*) → Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]
Fetch the next list of VC records from the store.

Parameters **max_count** – Max number of records to return. If not provided, defaults to the backend's preferred page size

Returns A list of *VCRecord* instances

aries_cloudagent.storage.vc_holder.in_memory module

Basic in-memory storage implementation of VC holder interface.

class aries_cloudagent.storage.vc_holder.in_memory.**InMemoryVCHolder**(*profile:* aries_cloudagent.core.in_memory.profile.InM

Bases: *aries_cloudagent.storage.vc_holder.base.VCHolder*

Basic in-memory storage class.

build_type_or_schema_query(*uri_list: Sequence[str]*) → dict
Build and return in-memory backend specific type_or_schema_query.

async delete_credential(*cred:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord)
Remove a previously-stored VC record.

Raises **StorageNotFoundError** – If the record is not found

async retrieve_credential_by_given_id(*given_id: str*) → *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*
Fetch a VC record by its given ID ('id' property).

Raises **StorageNotFoundError** – If the record is not found

async retrieve_credential_by_id(*record_id: str*) → *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*
Fetch a VC record by its record ID.

Raises **StorageNotFoundError** – If the record is not found

search_credentials(*contexts: Optional[Sequence[str]] = None*, *types: Optional[Sequence[str]] = None*, *schema_ids: Optional[str] = None*, *issuer_id: Optional[str] = None*, *subject_ids: Optional[str] = None*, *proof_types: Optional[Sequence[str]] = None*, *given_id: Optional[str] = None*, *tag_query: Optional[Mapping] = None*, *pd_uri_list: Optional[Sequence[str]] = None*) → *aries_cloudagent.storage.vc_holder.base.VCRecordSearch*
Start a new VC record search.

Parameters

- **contexts** – An inclusive list of JSON-LD contexts to match
- **types** – An inclusive list of JSON-LD types to match
- **schema_ids** – An inclusive list of credential schema identifiers
- **issuer_id** – The ID of the credential issuer
- **subject_ids** – The IDs of credential subjects all of which to match
- **proof_types** – The signature suite types used for the proof objects.
- **given_id** – The given id of the credential
- **tag_query** – A tag filter clause

async store_credential(*cred:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord)
Add a new VC record to the store.

Parameters **cred** – The VCRecord instance to store

Raises **StorageDuplicateError** – If the record_id is not unique

class aries_cloudagent.storage.vc_holder.in_memory.**InMemoryVCRecordSearch**(*search:* aries_cloudagent.storage.in_memory.I

Bases: *aries_cloudagent.storage.vc_holder.base.VCRecordSearch*

In-memory search for VC records.

async fetch(*max_count: Optional[int] = None*) → Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]

Fetch the next list of VC records from the store.

Parameters **max_count** – Max number of records to return. If not provided, defaults to the backend's preferred page size

Returns A list of *VCRecord* instances

aries_cloudagent.storage.vc_holder.indy module

Indy-SDK storage implementation of VC holder interface.

class aries_cloudagent.storage.vc_holder.indy.**IndySdkVCHolder**(*wallet:* aries_cloudagent.indy.sdk.wallet_setup.IndyOpenWall

Bases: *aries_cloudagent.storage.vc_holder.base.VCHolder*

Indy-SDK storage class.

build_type_or_schema_query(*uri_list: Sequence[str]*) → dict

Build and return indy-specific type_or_schema_query.

async delete_credential(*cred:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord)

Remove a previously-stored VC record.

Raises **StorageNotFoundError** – If the record is not found

async retrieve_credential_by_given_id(*given_id: str*) → *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*

Fetch a VC record by its given ID ('id' property).

Raises **StorageNotFoundError** – If the record is not found

async retrieve_credential_by_id(*record_id: str*) → *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*

Fetch a VC record by its record ID.

Raises **StorageNotFoundError** – If the record is not found

search_credentials(*contexts: Optional[Sequence[str]] = None, types: Optional[Sequence[str]] = None, schema_ids: Optional[Sequence[str]] = None, issuer_id: Optional[str] = None, subject_ids: Optional[str] = None, proof_types: Optional[Sequence[str]] = None, given_id: Optional[str] = None, tag_query: Optional[Mapping] = None, pd_uri_list: Optional[Sequence[str]] = None*) → *aries_cloudagent.storage.vc_holder.base.VCRecordSearch*

Start a new VC record search.

Parameters

- **contexts** – An inclusive list of JSON-LD contexts to match
- **types** – An inclusive list of JSON-LD types to match

- **schema_ids** – An inclusive list of credential schema identifiers
- **issuer_id** – The ID of the credential issuer
- **subject_ids** – The IDs of credential subjects all of which to match
- **proof_types** – The signature suite types used for the proof objects.
- **given_id** – The given id of the credential
- **tag_query** – A tag filter clause

async store_credential(*cred:* aries_cloudagent.storage.vc_holder.vc_record.VCRecord)
Add a new VC record to the store.

Parameters **cred** – The VCRecord instance to store

Raises **StorageDuplicateError** – If the record_id is not unique

class aries_cloudagent.storage.vc_holder.indy.**IndySdkVCRecordSearch**(*search:* aries_cloudagent.storage.indy.IndySdkStorage

Bases: *aries_cloudagent.storage.vc_holder.base.VCRecordSearch*

Indy-SDK storage search for VC records.

async close()
Dispose of the search query.

async fetch(*max_count: Optional[int] = None*) → Sequence[*aries_cloudagent.storage.vc_holder.vc_record.VCRecord*]
Fetch the next list of VC records from the store.

Parameters **max_count** – Max number of records to return. If not provided, defaults to the backend's preferred page size

Returns A list of *VCRecord* instances

aries_cloudagent.storage.vc_holder.vc_record module

Model for representing a stored verifiable credential.

class aries_cloudagent.storage.vc_holder.vc_record.**VCRecord**(*, *contexts: Sequence[str]*, *expanded_types: Sequence[str]*, *issuer_id: str*, *subject_ids: Sequence[str]*, *schema_ids: Sequence[str]*, *proof_types: Sequence[str]*, *cred_value: Mapping*, *given_id: Optional[str] = None*, *cred_tags: Optional[Mapping] = None*, *record_id: Optional[str] = None*)

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

Verifiable credential storage record class.

class Meta
Bases: object

VCRecord metadata.

schema_class = 'VCRecordSchema'

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

class aries_cloudagent.storage.vc_holder.vc_record.**VCRecordSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Verifiable credential storage record schema class.

class Meta
Bases: object

Verifiable credential storage record schema metadata.

model_class
alias of *aries_cloudagent.storage.vc_holder.vc_record.VCRecord*

contexts

cred_tags

cred_value

expanded_types

given_id

issuer_id

proof_types

record_id

schema_ids

subject_ids

aries_cloudagent.storage.vc_holder.xform module

Transformation between StorageRecord and VCRecord.

aries_cloudagent.storage.vc_holder.xform.**storage_to_vc_record**(*record:* aries_cloudagent.storage.record.StorageRecord) → *aries_cloudagent.storage.vc_holder.vc_record.VCReco*

Convert an Indy-SDK stored record into a VC record.

aries_cloudagent.storage.vc_holder.xform.**vc_to_storage_record**(*cred:* aries_cloudagent.storage.vc_holder.vc_record.VCRecc → *aries_cloudagent.storage.record.StorageRecord*

Convert a VC record into an in-memory stored record.

Submodules

aries_cloudagent.storage.askar module

aries_cloudagent.storage.base module

Abstract base classes for non-secrets storage.

class aries_cloudagent.storage.base.**BaseStorage**

Bases: abc.ABC

Abstract stored records interface.

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

Add a new record to the store.

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

abstract async delete_all_records(*type_filter: str*, *tag_query: Optional[Mapping] = None*)

Remove all records matching a particular type filter and tag query.

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

Delete an existing record.

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

abstract async find_all_records(*type_filter: str*, *tag_query: Optional[Mapping] = None*, *options: Optional[Mapping] = None*)

Retrieve all records matching a particular type filter and tag query.

async find_record(*type_filter: str*, *tag_query: Optional[Mapping] = None*, *options: Optional[Mapping] = None*) → *aries_cloudagent.storage.record.StorageRecord*

Find a record using a unique tag filter.

Parameters

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

abstract async get_record(*record_type: str*, *record_id: str*, *options: Optional[Mapping] = 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

abstract async update_record(*record:* aries_cloudagent.storage.record.StorageRecord, *value: str*, *tags: Mapping*)

Update an existing stored record's value and tags.

Parameters

- **record** – *StorageRecord* to update

- **value** – The new value
- **tags** – The new tags

class aries_cloudagent.storage.base.**BaseStorageSearch**

Bases: abc.ABC

Abstract stored records search interface.

abstract search_records(*type_filter: str*, *tag_query: Optional[Mapping] = None*, *page_size: Optional[int] = None*, *options: Optional[Mapping] = None*) → *aries_cloudagent.storage.base.BaseStorageSearchSession*

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 *BaseStorageSearchSession*

class aries_cloudagent.storage.base.**BaseStorageSearchSession**

Bases: abc.ABC

Abstract stored records search session interface.

async close()

Dispose of the search query.

abstract async fetch(*max_count: Optional[int] = None*) → Sequence[*aries_cloudagent.storage.record.StorageRecord*]

Fetch the next list of results from the store.

Parameters **max_count** – Max number of records to return. If not provided, defaults to the backend's preferred page size

Returns A list of *StorageRecord* instances

class aries_cloudagent.storage.base.**IterSearch**(*search:* aries_cloudagent.storage.base.BaseStorageSearchSession, *page_size: Optional[int] = None*)

Bases: object

A generic record search async iterator.

aries_cloudagent.storage.base.**validate_record**(*record:* aries_cloudagent.storage.record.StorageRecord, **, delete=False*)

Ensure that a record is ready to be saved/updated/deleted.

aries_cloudagent.storage.error module

Storage-related exceptions.

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

Bases: *aries_cloudagent.storage.error.StorageError*

Duplicate record found in storage.

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

Bases: *aries_cloudagent.core.error.BaseError*

Base class for Storage errors.

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

Bases: *aries_cloudagent.storage.error.StorageError*

Record not found in storage.

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

Bases: *aries_cloudagent.storage.error.StorageError*

General exception during record search.

aries_cloudagent.storage.in_memory module

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

class aries_cloudagent.storage.in_memory.**InMemoryStorage**(*profile:* aries_cloudagent.core.in_memory.profile.InMemoryProfile)

Bases: *aries_cloudagent.storage.base.BaseStorage*, *aries_cloudagent.storage.base.BaseStorageSearch*

Basic in-memory storage class.

async 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

async delete_all_records(*type_filter: str*, *tag_query: Optional[Mapping] = None*)

Remove all records matching a particular type filter and tag query.

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

Delete a record.

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

Raises **StorageNotFoundError** – If record not found

async find_all_records(*type_filter: str*, *tag_query: Optional[Mapping] = None*, *options: Optional[Mapping] = None*)

Retrieve all records matching a particular type filter and tag query.

async get_record(*record_type: str*, *record_id: str*, *options: Optional[Mapping] = 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: Optional[Mapping] = None*, *page_size: Optional[int] = None*, *options: Optional[Mapping] = None*) → *aries_cloudagent.storage.in_memory.InMemoryStorageSearch*

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 *InMemoryStorageSearch*

async update_record(*record:* aries_cloudagent.storage.record.StorageRecord, *value: str*, *tags: Mapping*)

Update an existing stored record's value.

Parameters

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

Raises **StorageNotFoundError** – If record not found

class aries_cloudagent.storage.in_memory.**InMemoryStorageSearch**(*profile:* aries_cloudagent.core.in_memory.profile.InMemoryl *type_filter: str*, *tag_query: Mapping*, *page_size: Optional[int] = None*, *options: Optional[Mapping] = None*)

Bases: *aries_cloudagent.storage.base.BaseStorageSearchSession*

Represent an active stored records search.

async close()

Dispose of the search query.

async fetch(*max_count: Optional[int] = None*) → Sequence[*aries_cloudagent.storage.record.StorageRecord*]

Fetch the next list of results from the store.

Parameters **max_count** – Max number of records to return. If not provided, defaults to the backend's preferred page size

Returns A list of *StorageRecord* instances

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

aries_cloudagent.storage.in_memory.**tag_query_match**(*tags: dict*, *tag_query: dict*) → bool
Match simple tag filters (string values).

aries_cloudagent.storage.in_memory.**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.indy module

Indy implementation of BaseStorage interface.

class aries_cloudagent.storage.indy.**IndySdkStorage**(*wallet:* aries_cloudagent.indy.sdk.wallet_setup.IndyOpenWallet)
Bases: *aries_cloudagent.storage.base.BaseStorage*, *aries_cloudagent.storage.base.BaseStorageSearch*

Indy Non-Secrets interface.

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

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

async delete_all_records(*type_filter: str*, *tag_query: Optional[Mapping] = None*)
Remove all records matching a particular type filter and tag query.

async 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

async find_all_records(*type_filter: str*, *tag_query: Optional[Mapping] = None*, *options: Optional[Mapping] = None*)
Retrieve all records matching a particular type filter and tag query.

async get_record(*record_type: str*, *record_id: str*, *options: Optional[Mapping] = 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: Optional[Mapping] = None*, *page_size: Optional[int] = None*, *options: Optional[Mapping] = None*) → *aries_cloudagent.storage.indy.IndySdkStorageSearch*

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 *IndySdkStorageSearch*

async update_record(*record:* aries_cloudagent.storage.record.StorageRecord, *value: str*, *tags: Mapping*)

Update an existing stored record's value and tags.

Parameters

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

Raises

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

property wallet: *aries_cloudagent.indy.sdk.wallet_setup.IndyOpenWallet*

Accessor for IndyOpenWallet instance.

class aries_cloudagent.storage.indy.**IndySdkStorageSearch**(*store:* aries_cloudagent.storage.indy.IndySdkStorage, *type_filter: str*, *tag_query: Mapping*, *page_size: Optional[int] = None*, *options: Optional[Mapping] = None*)

Bases: *aries_cloudagent.storage.base.BaseStorageSearchSession*

Represent an active stored records search.

async close()

Dispose of the search query.

async fetch(*max_count: Optional[int] = None*) → Sequence[*aries_cloudagent.storage.record.StorageRecord*]

Fetch the next list of results from the store.

Parameters **max_count** – Max number of records to return. If not provided, defaults to the backend's preferred page size

Returns A list of *StorageRecord* instances

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

aries_cloudagent.storage.record module

Record instance stored and searchable by BaseStorage implementation.

class aries_cloudagent.storage.record.**StorageRecord**(*type*, *value*, *tags: Optional[dict] = None*, *id: Optional[str] = None*)

Bases: *aries_cloudagent.storage.record.StorageRecord*

Storage record class.

aries_cloudagent.tails package

Submodules

aries_cloudagent.tails.base module

Tails server interface base class.

class aries_cloudagent.tails.base.**BaseTailsServer**

Bases: abc.ABC

Base class for tails server interface.

abstract async upload_tails_file(*context: aries_cloudagent.config.injection_context.InjectionContext*, *rev_reg_id: str*, *tails_file_path: str*, *interval: float = 1.0*, *backoff: float = 0.25*, *max_attempts: int = 5*) → Tuple[bool, str]

Upload tails file to tails server.

Parameters

- **rev_reg_id** – The revocation registry identifier
- **tails_file** – The path to the tails file to upload
- **interval** – initial interval between attempts
- **backoff** – exponential backoff in retry interval
- **max_attempts** – maximum number of attempts to make

aries_cloudagent.tails.error module

Tails server related errors.

exception aries_cloudagent.tails.error.**TailsServerNotConfiguredError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Error indicating the tails server plugin hasn't been configured.

aries_cloudagent.tails.indy_tails_server module

Indy tails server interface class.

class aries_cloudagent.tails.indy_tails_server.**IndyTailsServer**
Bases: *aries_cloudagent.tails.base.BaseTailsServer*

Indy tails server interface.

async upload_tails_file(*context:* aries_cloudagent.config.injection_context.InjectionContext, *rev_reg_id: str*, *tails_file_path: str*, *interval: float = 1.0*, *backoff: float = 0.25*, *max_attempts: int = 5*) → Tuple[bool, str]
Upload tails file to tails server.

Parameters

- **context** – context with configuration settings
- **rev_reg_id** – revocation registry identifier
- **tails_file_path** – path to the tails file to upload
- **interval** – initial interval between attempts
- **backoff** – exponential backoff in retry interval
- **max_attempts** – maximum number of attempts to make

aries_cloudagent.transport package

Subpackages

aries_cloudagent.transport.inbound package

Submodules

aries_cloudagent.transport.inbound.base module

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.outbound.messa

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

aries_cloudagent.transport.inbound.manager module

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.receipt.Message*, **, connection_id: Optional[str] = None*, *session_id: Optional[str] = None*, *transport_type: Optional[str] = None*)

Bases: object

Container class linking a message payload with its receipt details.

dispatch_processing_complete()
Dispatch processing complete.

async wait_processing_complete()
Wait for processing to complete.

aries_cloudagent.transport.inbound.receipt module

Classes for representing message receipt details.

class aries_cloudagent.transport.inbound.receipt.**MessageReceipt**(**, connection_id: Optional[str] = None*, *direct_response_mode: Optional[str] = None*, *in_time: Optional[datetime.datetime] = None*, *raw_message: Optional[str] = None*, *recipient_verkey: Optional[str] = None*, *recipient_did: Optional[str] = None*, *recipient_did_public: Optional[bool] = None*, *sender_did: Optional[str] = None*, *sender_verkey: Optional[str] = None*, *thread_id: Optional[str] = None*, *parent_thread_id: Optional[str] = None*)

Bases: object

Properties of an agent message's delivery.

REPLY_MODE_ALL = 'all'

REPLY_MODE_NONE = 'none'

REPLY_MODE_THREAD = 'thread'

property connection_id: str
Accessor for the pairwise connection identifier.

Returns This context's connection identifier

property direct_response_mode: str
Accessor for the requested direct response mode.

Returns This context's requested direct response mode

property direct_response_requested: str
Accessor for the the state of the direct response mode.

Returns This context's requested direct response mode

property in_time: str
Accessor for the datetime the message was received.

Returns This context's received time

property parent_thread_id: Optional[str]
Accessor for the identifier of the message parent thread.

Returns The delivery parent thread ID

property raw_message: str
Accessor for the raw message text.

Returns The raw message text

property recipient_did: str
Accessor for the recipient DID which corresponds with the verkey.

Returns The recipient DID

property recipient_did_public: bool
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

property recipient_verkey: str
Accessor for the recipient verkey key used to pack the incoming request.

Returns The recipient verkey

property sender_did: str
Accessor for the sender DID which corresponds with the verkey.

Returns The sender did

property sender_verkey: str
Accessor for the sender public key used to pack the incoming request.

Returns This context's sender's verkey

property thread_id: str
Accessor for the identifier of the message thread.

Returns The delivery thread ID

aries_cloudagent.transport.inbound.session module

aries_cloudagent.transport.inbound.ws module

aries_cloudagent.transport.outbound package

Submodules

aries_cloudagent.transport.outbound.base module

Base outbound transport.

class aries_cloudagent.transport.outbound.base.**BaseOutboundTransport**(*wire_format: Optional[aries_cloudagent.transport.wire_forma = None*, *root_profile: Optional[aries_cloudagent.core.profile.Profile] = None*)

Bases: abc.ABC

Base outbound transport class.

property collector: *aries_cloudagent.utils.stats.Collector*
Accessor for the stats collector instance.

abstract async handle_message(*profile:* aries_cloudagent.core.profile.Profile, *payload: Union[str, bytes]*, *endpoint: str*, *metadata: Optional[dict] = None*)
Handle message.

Parameters

- **profile** – the profile that produced the message
- **payload** – message payload in string or byte format
- **endpoint** – URI endpoint for delivery
- **metadata** – Additional metadata associated with the payload

abstract async start()
Start the transport.

abstract async stop()
Shut down the transport.

property wire_format: *aries_cloudagent.transport.wire_format.BaseWireFormat*
Accessor for a custom wire format for the transport.

exception aries_cloudagent.transport.outbound.base.**OutboundDeliveryError**(**args*, *error_code: Optional[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: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.transport.error.TransportError*

Generic outbound transport error.

exception aries_cloudagent.transport.outbound.base.**OutboundTransportRegistrationError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

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**(***kwargs*)

Bases: *aries_cloudagent.transport.outbound.base.BaseOutboundTransport*

Http outbound transport class.

async handle_message(*profile:* aries_cloudagent.core.profile.Profile, *payload: Union[str, bytes]*, *endpoint: str*, *metadata: Optional[dict] = None*, *api_key: Optional[str] = None*)

Handle message from queue.

Parameters

- **profile** – the profile that produced the message
- **payload** – message payload in string or byte format
- **endpoint** – URI endpoint for delivery
- **metadata** – Additional metadata associated with the payload

is_external = False

schemes = ('http', 'https')

async start()

Start the transport.

async stop()

Stop the transport.

aries_cloudagent.transport.outbound.manager module

Outbound transport manager.

class aries_cloudagent.transport.outbound.manager.**OutboundTransportManager**(*profile:* aries_cloudagent.core.profile.Profile, *handle_not_delivered: Optional[Callable] = None*)

Bases: object

Outbound transport manager class.

MAX_RETRY_COUNT = 4

deliver_queued_message(*queued:* aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage) → _asyncio.Task

Kick off delivery of a queued message.

async encode_outbound_message(*profile:* aries_cloudagent.core.profile.Profile, *outbound:* aries_cloudagent.transport.outbound.message.OutboundMessage, *target:* aries_cloudagent.connections.models.connection_target.ConnectionTarget)

Encode outbound message for the target.

Parameters

- **profile** – The active profile for the request
- **outbound** – The outbound message to deliver
- **target** – The outbound message target

encode_queued_message(*queued:* aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage) → _asyncio.Task

Kick off encoding of a queued message.

async enqueue_message(*profile:* aries_cloudagent.core.profile.Profile, *outbound:* aries_cloudagent.transport.outbound.message.OutboundMessage)

Add an outbound message to the queue.

Parameters

- **profile** – The active profile for the request
- **outbound** – The outbound message to deliver

enqueue_webhook(*topic: str*, *payload: dict*, *endpoint: str*, *max_attempts: Optional[int] = None*, *metadata: Optional[dict] = 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
- **metadata** – Additional metadata associated with the payload

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.

async flush()

Wait for any queued messages to be delivered.

get_external_running_transport() → str

Find the external running transport ID.

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.

async perform_encode(*queued:* aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage, *wire_format: Optional[*aries_cloudagent.transport.wire_format.BaseWireFormat*]* = *None*)
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_name: str*) → str
Register a new outbound transport by module path.

Parameters **module_name** – 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: Optional[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

async setup()
Perform setup operations.

async start()
Start all transports and feed messages from the queue.

async start_transport(*transport_id: str*)
Start a registered transport.

async stop(*wait: bool* = *True*)
Stop all running transports.

class aries_cloudagent.transport.outbound.manager.**QueuedOutboundMessage**(*profile:* aries_cloudagent.core.profile.Profile, *message:* aries_cloudagent.transport.outbound.mes *target:* aries_cloudagent.connections.models.cor *transport_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: Optional[str] = None*, *enc_payload: Optional[Union[str, bytes]] = None*, *endpoint: Optional[str] = None*, *payload: Union[str, bytes]*, *reply_session_id: Optional[str] = None*, *reply_thread_id: Optional[str] = None*, *reply_to_verkey: Optional[str] = None*, *reply_from_verkey: Optional[str] = None*, *target: Optional[*aries_cloudagent.connections.models.conn *= None*, *target_list: Optional[Sequence[*aries_cloudagent.connections.m *= None*, *to_session_only: bool = False*)

Bases: object

Represents an outgoing message.

aries_cloudagent.transport.outbound.status module

Enum representing captured send status of outbound messages.

class aries_cloudagent.transport.outbound.status.**OutboundSendStatus**(*value*)
Bases: enum.Enum

Send status of outbound messages.

QUEUED_FOR_DELIVERY = 'queued_for_delivery'

SENT_TO_EXTERNAL_QUEUE = 'sent_to_external_queue'

SENT_TO_SESSION = 'sent_to_session'

UNDELIVERABLE = 'undeliverable'

WAITING_FOR_PICKUP = 'waiting_for_pickup'

property topic
Return an event topic associated with a given status.

aries_cloudagent.transport.outbound.ws module

Websockets outbound transport.

class aries_cloudagent.transport.outbound.ws.**WsTransport**(***kwargs*)
Bases: *aries_cloudagent.transport.outbound.base.BaseOutboundTransport*

Websockets outbound transport class.

async handle_message(*profile:* aries_cloudagent.core.profile.Profile, *payload: Union[str, bytes]*, *endpoint: str*, *metadata: Optional[dict] = None*, *api_key: Optional[str] = None*)
Handle message from queue.

Parameters

- **profile** – the profile that produced the message
- **payload** – message payload in string or byte format
- **endpoint** – URI endpoint for delivery
- **metadata** – Additional metadata associated with the payload

is_external = False

schemes = ('ws', 'wss')

async start()
Start the outbound transport.

async 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.

abstract async dequeue(*, *timeout: Optional[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** –

abstract async 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** –

abstract async join()
Wait for the queue to empty.

abstract reset()
Empty the queue and reset the stop event.

abstract stop()
Cancel active iteration of the queue.

abstract 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.

async dequeue(*, *timeout: Optional[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** –

async 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** –

async 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.**RecipientKeysError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.transport.error.WireFormatError*

Extract recipient keys error.

exception aries_cloudagent.transport.error.**TransportError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Base class for all transport errors.

exception aries_cloudagent.transport.error.**WireFormatEncodeError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.transport.error.WireFormatError*

Encoding error when packing the wire format.

error_code = 'message_encode_error'

exception aries_cloudagent.transport.error.**WireFormatError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.transport.error.TransportError*

Base class for wire-format errors.

exception aries_cloudagent.transport.error.**WireFormatParseError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.transport.error.WireFormatError*

Parse error when unpacking the wire format.

error_code = 'message_parse_error'

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.

async encode_message(*session:* aries_cloudagent.core.profile.ProfileSession, *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

- **session** – The profile session for providing wallet access
- **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

get_recipient_keys(*message_body: Union[str, bytes]*) → List[str]

Get all recipient keys from a wire message.

Parameters **message_body** – The body of the message

Returns List of recipient keys from the message body

Raises **RecipientKeysError** – If the recipient keys could not be extracted

async pack(*session:* aries_cloudagent.core.profile.ProfileSession, *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.

async parse_message(*session:* aries_cloudagent.core.profile.ProfileSession, *message_body: Union[str, bytes]*) → Tuple[dict, *aries_cloudagent.transport.inbound.receipt.MessageReceipt*]

Deserialize an incoming message and further populate the request context.

Parameters

- **session** – The profile session for providing wallet access
- **message_body** – The body of the message

Returns A tuple of the parsed message and a message receipt instance

Raises

- **WireFormatParseError** – If the JSON parsing failed
- **WireFormatParseError** – If a wallet is required but can't be located

async unpack(*session:* aries_cloudagent.core.profile.ProfileSession, *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**(**args: Any*, ***kwargs: Any*)
Bases: aiohttp.

Attach hooks to client session events and report statistics.

async connection_queued_end(*session*, *context*, *params*)
Handle the end of a queued connection.

async connection_queued_start(*session*, *context*, *params*)
Handle the start of a queued connection.

async connection_ready(*session*, *context*, *params*)
Handle the end of connection acquisition.

async dns_resolvehost_end(*session*, *context*, *params*)
Handle the end of a DNS resolution.

async dns_resolvehost_start(*session*, *context*, *params*)
Handle the start of a DNS resolution.

async request_end(*session*, *context*, *params*)
Handle the end of request.

async request_start(*session*, *context*, *params*)
Handle the start of a request.

async 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.

abstract async encode_message(*session:* aries_cloudagent.core.profile.ProfileSession, *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

- **session** – The profile session for providing wallet access
- **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

abstract get_recipient_keys(*message_body: Union[str, bytes]*) → List[str]
Get all recipient keys from a wire message.

Parameters **message_body** – The body of the message

Returns List of recipient keys from the message body

Raises **RecipientKeysError** – If the recipient keys could not be extracted

abstract async parse_message(*session:* aries_cloudagent.core.profile.ProfileSession, *message_body: Union[str, bytes]*) → Tuple[dict, *aries_cloudagent.transport.inbound.receipt.MessageReceipt*]
Deserialize an incoming message and further populate the request context.

Parameters

- **session** – The profile session for providing wallet access
- **message_body** – The body of the message

Returns A tuple of the parsed message and a message receipt instance

Raises **WireFormatParseError** – If the message can't be parsed

class aries_cloudagent.transport.wire_format.**JsonWireFormat**
Bases: *aries_cloudagent.transport.wire_format.BaseWireFormat*

Unencrypted wire format.

abstract async encode_message(*session:* aries_cloudagent.core.profile.ProfileSession, *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

- **session** – The profile session for providing wallet access
- **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

get_recipient_keys(*message_body: Union[str, bytes]*) → List[str]
Get all recipient keys from a wire message.

Parameters **message_body** – The body of the message

Returns List of recipient keys from the message body

Raises **RecipientKeysError** – If the recipient keys could not be extracted

abstract async parse_message(*session:* aries_cloudagent.core.profile.ProfileSession, *message_body: Union[str, bytes]*) → Tuple[dict, *aries_cloudagent.transport.inbound.receipt.MessageReceipt*]
Deserialize an incoming message and further populate the request context.

Parameters

- **session** – The profile session for providing wallet access

- **message_body** – The body of the message

Returns A tuple of the parsed message and a message receipt instance

Raises **WireFormatParseError** – If the JSON parsing failed

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: Optional[str] = None*, *package: Optional[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: Optional[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*, *mod_path: str*, *package: Optional[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: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Class not found error.

class aries_cloudagent.utils.classloader.**DeferLoad**(*cls_path: str*)
Bases: object

Helper to defer loading of a class definition.

property resolved
Accessor for the resolved class instance.

exception aries_cloudagent.utils.classloader.**ModuleLoadError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Module load error.

aries_cloudagent.utils.dependencies module

Dependency related util methods.

aries_cloudagent.utils.dependencies.**assert_ursa_bbs_signatures_installed**()
Assert ursa_bbs_signatures module is installed.

aries_cloudagent.utils.dependencies.**is_indy_sdk_module_installed**()
Check whether indy (indy-sdk) module is installed.

Returns Whether indy (indy-sdk) is installed.

Return type bool

aries_cloudagent.utils.dependencies.**is_ursa_bbs_signatures_module_installed**()
Check whether ursa_bbs_signatures module is installed.

Returns Whether ursa_bbs_signatures is installed.

Return type bool

aries_cloudagent.utils.env module

Environment utility methods.

aries_cloudagent.utils.env.**storage_path**(**subpaths*, *create: bool = False*) → pathlib.Path
Get the default aca-py home directory.

aries_cloudagent.utils.http module

HTTP utility methods.

exception aries_cloudagent.utils.http.**FetchError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Error raised when an HTTP fetch fails.

exception aries_cloudagent.utils.http.**PutError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)

Bases: *aries_cloudagent.core.error.BaseError*

Error raised when an HTTP put fails.

async aries_cloudagent.utils.http.**fetch**(*url: str*, ***, *headers: Optional[dict] = None*, *retry: bool = True*, *max_attempts: int = 5*, *interval: float = 1.0*, *backoff: float = 0.25*, *request_timeout: float = 10.0*, *connector: Optional[aiohttp.BaseConnector] = None*, *session: Optional[aiohttp.ClientSession] = 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

async aries_cloudagent.utils.http.**fetch_stream**(*url: str*, ***, *headers: Optional[dict] = None*, *retry: bool = True*, *max_attempts: int = 5*, *interval: float = 1.0*, *backoff: float = 0.25*, *request_timeout: float = 10.0*, *connector: Optional[aiohttp.BaseConnector] = None*, *session: Optional[aiohttp.ClientSession] = 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

async aries_cloudagent.utils.http.**put_file**(*url: str*, *file_data: dict*, *extra_data: dict*, ***, *retry: bool = True*, *max_attempts: int = 5*, *interval: float = 1.0*, *backoff: float = 0.25*, *request_timeout: float = 10.0*, *connector: Optional[aiohttp.BaseConnector] = None*, *session: Optional[aiohttp.ClientSession] = None*, *json: bool = False*)

Put to HTTP server with automatic retries and timeouts.

Parameters

- **url** – the address to use
- **file_data** – dict with data key and path of file to upload
- **extra_data** – further content to include in data to put
- **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.jwe module

JSON Web Encryption utilities.

class aries_cloudagent.utils.jwe.**B64Value**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.fields.

A marshmallow-compatible wrapper for base64-URL values.

class aries_cloudagent.utils.jwe.**JweEnvelope**(***, *protected: Optional[dict] = None*, *protected_b64: Optional[bytes] = None*, *unprotected: Optional[dict] = None*, *ciphertext: Optional[bytes] = None*, *iv: Optional[bytes] = None*, *tag: Optional[bytes] = None*, *aad: Optional[bytes] = None*, *with_protected_recipients: bool = False*, *with_flatten_recipients: bool = True*)

Bases: object

JWE envelope instance.

add_recipient(*recip:* aries_cloudagent.utils.jwe.JweRecipient)

Add a recipient to the JWE envelope.

property combined_aad: bytes

Accessor for the additional authenticated data.

classmethod deserialize(*message: Mapping[str, Any]*) → *aries_cloudagent.utils.jwe.JweEnvelope*
Deserialize a JWE envelope from a mapping.

classmethod from_json(*message: Union[bytes, str]*) → *aries_cloudagent.utils.jwe.JweEnvelope*
Decode a JWE envelope from a JSON string or bytes value.

get_recipient(*kid: str*) → *aries_cloudagent.utils.jwe.JweRecipient*
Find a recipient by key ID.

property protected_bytes: bytes
Access the protected data encoded as bytes.

This value is used in the additional authenticated data when encrypting.

property recipient_key_ids: Iterable[*aries_cloudagent.utils.jwe.JweRecipient*]
Accessor for an iterator over the JWE recipient key identifiers.

property recipients: Iterable[*aries_cloudagent.utils.jwe.JweRecipient*]
Accessor for an iterator over the JWE recipients.

The headers for each recipient include protected and unprotected headers from the outer envelope.

property recipients_json: List[Dict[str, Any]]
Encode the current recipients for JSON.

serialize() → dict
Serialize the JWE envelope to a mapping.

set_payload(*ciphertext: bytes*, *iv: bytes*, *tag: bytes*, *aad: Optional[bytes] = None*)
Set the payload of the JWE envelope.

set_protected(*protected: Mapping[str, Any]*)
Set the protected headers of the JWE envelope.

to_json() → str
Serialize the JWE envelope to a JSON string.

class aries_cloudagent.utils.jwe.**JweRecipient**(*, *encrypted_key: bytes*, *header: Optional[dict] = None*)
Bases: object

A single message recipient.

classmethod deserialize(*entry: Mapping[str, Any]*) → *aries_cloudagent.utils.jwe.JweRecipient*
Deserialize a JWE recipient from a mapping.

serialize() → dict
Serialize the JWE recipient to a mapping.

class aries_cloudagent.utils.jwe.**JweRecipientSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

JWE recipient schema.

encrypted_key
A marshmallow-compatible wrapper for base64-URL values.

header

class aries_cloudagent.utils.jwe.**JweSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

JWE envelope schema.

aad
A marshmallow-compatible wrapper for base64-URL values.

ciphertext
A marshmallow-compatible wrapper for base64-URL values.

encrypted_key
A marshmallow-compatible wrapper for base64-URL values.

header

iv
A marshmallow-compatible wrapper for base64-URL values.

protected

recipients

tag
A marshmallow-compatible wrapper for base64-URL values.

unprotected

aries_cloudagent.utils.jwe.**b64url**(*value: Union[bytes, str]*) → str
Encode a string or bytes value as unpadded base64-URL.

aries_cloudagent.utils.jwe.**from_b64url**(*value: str*) → bytes
Decode an unpadded base64-URL value.

aries_cloudagent.utils.outofband module

Utilities for creating out-of-band messages.

aries_cloudagent.utils.outofband.**serialize_outofband**(*message:* aries_cloudagent.messaging.agent_message.AgentMessage, *did:* aries_cloudagent.wallet.did_info.DIDInfo, *endpoint: str*) → str
Serialize the agent message as an out-of-band message.

Returns An OOB message in URL format.

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.

property final: bool
Check if this is the last instance in the sequence.

next() → *aries_cloudagent.utils.repeat.RepeatAttempt*
Get the next attempt instance.

property next_interval: float
Calculate the interval before the next attempt.

timeout(*interval: Optional[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: Optional[str] = None*)

Bases: object

Collector for a set of statistics.

property enabled: bool
Accessor for the collector's enabled property.

extract(*groups: Optional[Sequence[str]] = None*) → dict
Extract statistics for a specific set of groups.

log(*name: str*, *duration: float*, *start: Optional[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.

property results: dict
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: Optional[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: Optional[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: Optional[str] = None*, *timing: Optional[dict] = None*)

Bases: object

Represent the result of a queued task.

class aries_cloudagent.utils.task_queue.**PendingTask**(*coro: Coroutine*, *complete_hook: Optional[Callable] = None*, *ident: Optional[str] = None*, *task_future: Optional[_asyncio.Future] = None*, *queued_time: Optional[float] = None*)

Bases: object

Represent a task in the queue.

cancel()
Cancel the pending task.

property cancelled
Accessor for the cancelled property.

property task: _asyncio.Task
Accessor for the task.

class aries_cloudagent.utils.task_queue.**TaskQueue**(*max_active: int = 0*, *timed: bool = False*, *trace_fn: Optional[Callable] = None*)

Bases: object

A class for managing a set of asyncio tasks.

add_active(*task: _asyncio.Task*, *task_complete: Optional[Callable] = None*, *ident: Optional[str] = None*, *timing: Optional[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.

property cancelled: bool
Accessor for the cancelled property of the queue.

async complete(*timeout: Optional[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: Optional[dict] = None*)
Clean up after a task has completed and run callbacks.

property current_active: int
Accessor for the current number of active tasks in the queue.

property current_pending: int
Accessor for the current number of pending tasks in the queue.

property current_size: int
Accessor for the total number of tasks in the queue.

drain() → _asyncio.Task
Start the process to run queued tasks.

async flush()
Wait for any active or pending tasks to be completed.

property max_active: int
Accessor for the maximum number of active tasks in the queue.

put(*coro: Coroutine*, *task_complete: Optional[Callable] = None*, *ident: Optional[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

property ready: bool
Accessor for the ready property of the queue.

run(*coro: Coroutine*, *task_complete: Optional[Callable] = None*, *ident: Optional[str] = None*, *timing: Optional[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

async wait_for(*timeout: float*)
Wait for all queued tasks to complete with a timeout.

aries_cloudagent.utils.task_queue.**coro_ident**(*coro: Coroutine*)
Extract an identifier for a coroutine.

async aries_cloudagent.utils.task_queue.**coro_timed**(*coro: Coroutine*, *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.

aries_cloudagent.utils.tracing module

Event tracing.

class aries_cloudagent.utils.tracing.**AdminAPIMessageTracingSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Request/result schema including agent message tracing.

This is to be used as a superclass for aca-py admin input/output messages that need to support tracing.

trace

aries_cloudagent.utils.tracing.**decode_inbound_message**(*message*)
Return bundled message if appropriate.

aries_cloudagent.utils.tracing.**get_timer**() → float
Return a timer.

aries_cloudagent.utils.tracing.**trace_event**(*context*, *message*, *handler: Optional[str] = None*, *outcome: Optional[str] = None*, *perf_counter: Optional[float] = None*, *force_trace: bool = False*, *raise_errors: bool = False*) → float
Log a trace event to a configured target.

Parameters

- **context** – The application context, attributes of interest are: context["trace.enabled"]: True if we are logging events context["trace.target"]: Trace target ("log", "message" or an http endpoint) context["trace.tag"]: Tag to be included in trace output
- **message** – the current message, can be an AgentMessage, InboundMessage, OutboundMessage or Exchange record
- **event** – Dict that will be converted to json and posted to the target

aries_cloudagent.utils.tracing.**tracing_enabled**(*context*, *message*) → bool
Determine whether to log trace messages or not.

aries_cloudagent.vc package

Subpackages

aries_cloudagent.vc.ld_proofs package

class aries_cloudagent.vc.ld_proofs.**AssertionProofPurpose**(*, *date: Optional[datetime.datetime] = None*, *max_timestamp_delta: Optional[datetime.timedelta] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.purposes.controller_proof_purpose.ControllerProofPurpose*

Assertion proof purpose class.

term = 'assertionMethod'

class aries_cloudagent.vc.ld_proofs.**AuthenticationProofPurpose**(*, *challenge: str*, *domain: Optional[str] = None*, *date: Optional[datetime.datetime] = None*, *max_timestamp_delta: Optional[datetime.timedelta] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.purposes.controller_proof_purpose.ControllerProofPurpose*

Authentication proof purpose.

term = 'authentication'

update(*proof: dict*) → dict

Update poof purpose, challenge and domain on proof.

validate(*, *proof: dict*, *document: dict*, *suite:* LinkedDataProof, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*

Validate whether challenge and domain are valid.

class aries_cloudagent.vc.ld_proofs.**BbsBlsSignature2020**(*, *key_pair:* aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair, *proof: Optional[dict] = None*, *verification_method: Optional[str] = None*, *date: Optional[datetime.datetime] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020_base.BbsBlsSignature2020Base*

BbsBlsSignature2020 class.

async create_proof(*, *document: dict*, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → dict

Create proof for document, return proof.

async sign(*, *verify_data: List[bytes]*, *proof: dict*) → dict

Sign the data and add it to the proof.

Parameters

- **verify_data** (*List[bytes]*) – The data to sign.
- **proof** (*dict*) – The proof to add the signature to

Returns The proof object with the added signature

Return type dict

signature_type = 'BbsBlsSignature2020'

async verify_proof(*, *proof: dict*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*

Verify proof against document and proof purpose.

async verify_signature(*, *verify_data: List[bytes]*, *verification_method: dict*, *document: dict*, *proof: dict*, *document_loader: Callable[[str, dict], dict]*) → bool

Verify the data against the proof.

Parameters

- **verify_data** (*bytes*) – The data to check
- **verification_method** (*dict*) – The verification method to use.
- **document** (*dict*) – The document the verify data is derived for as extra context
- **proof** (*dict*) – The proof to check
- **document_loader** (DocumentLoader) – Document loader used for resolving

Returns Whether the signature is valid for the data

Return type bool

class aries_cloudagent.vc.ld_proofs.**BbsBlsSignatureProof2020**(*, *key_pair: aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020_base.BbsBlsSignature2020Base*

BbsBlsSignatureProof2020 class.

async derive_proof(*, *proof: dict*, *document: dict*, *reveal_document: dict*, *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*)

Derive proof for document, return dict with derived document and proof.

signature_type = 'BbsBlsSignatureProof2020'

supported_derive_proof_types = ['BbsBlsSignature2020', 'sec:BbsBlsSignature2020', 'https://w3id.org/security#BbsBlsSignature2020']

async verify_proof(*, *proof: dict*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*

Verify proof against document and proof purpose.

class aries_cloudagent.vc.ld_proofs.**ControllerProofPurpose**(*, *term: str*, *date: Optional[datetime.datetime] = None*, *max_timestamp_delta: Optional[datetime.timedelta] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*

Controller proof purpose class.

validate(*, *proof: dict*, *document: dict*, *suite:* LinkedDataProof, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*

Validate whether verification method of proof is authorized by controller.

class aries_cloudagent.vc.ld_proofs.**CredentialIssuancePurpose**(*, *date: Optional[datetime.datetime] = None*, *max_timestamp_delta: Optional[datetime.timedelta] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.purposes.assertion_proof_purpose.AssertionProofPurpose*

Credential Issuance proof purpose.

validate(*, *proof: dict*, *document: dict*, *suite:* LinkedDataProof, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*

Validate if the issuer matches the controller of the verification method.

class aries_cloudagent.vc.ld_proofs.**DocumentLoader**(*profile:* aries_cloudagent.core.profile.Profile, *cache_ttl: int = 300*)

Bases: object

JSON-LD document loader.

async load_document(*url: str*, *options: dict*)

Load JSON-LD document.

Method signature conforms to PyLD document loader interface

Document loading is processed in separate thread to deal with async to sync transformation.

class aries_cloudagent.vc.ld_proofs.**DocumentVerificationResult**(*, *verified: bool*, *document: Optional[dict] = None*, *results: Optional[List[*aries_cloudagent.vc.ld_proofs.validation_ *= None*, *errors: Optional[List[Exception]] = None*)

Bases: object

Domain verification result class.

class aries_cloudagent.vc.ld_proofs.**Ed25519Signature2018**(*, *key_pair:* aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair, *proof: Optional[dict] = None*, *verification_method: Optional[str] = None*, *date: Optional[Union[datetime.datetime, str]] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.jws_linked_data_signature.JwsLinkedDataSignature*

Ed25519Signature2018 suite.

signature_type = 'Ed25519Signature2018'

class aries_cloudagent.vc.ld_proofs.**JwsLinkedDataSignature**(*, *signature_type: str*, *algorithm: str*, *required_key_type: str*, *key_pair: aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair*, *proof: Optional[dict] = None*, *verification_method: Optional[str] = None*, *date: Optional[Union[datetime.datetime, str]] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.linked_data_signature.LinkedDataSignature*

JWS Linked Data class.

async sign(*, *verify_data: bytes*, *proof: dict*) → dict

Sign the data and add it to the proof.

Adds a jws to the proof that can be used for multiple signature algorithms.

Parameters

- **verify_data** (*bytes*) – The data to sign.
- **proof** (*dict*) – The proof to add the signature to

Returns The proof object with the added signature

Return type dict

async verify_signature(*, *verify_data: bytes*, *verification_method: dict*, *document: dict*, *proof: dict*, *document_loader: Callable[[str, dict], dict]*)

Verify the data against the proof.

Checks for a jws on the proof.

Parameters

- **verify_data** (*bytes*) – The data to check
- **verification_method** (*dict*) – The verification method to use.
- **document** (*dict*) – The document the verify data is derived for as extra context
- **proof** (*dict*) – The proof to check
- **document_loader** (DocumentLoader) – Document loader used for resolving

Returns Whether the signature is valid for the data

Return type bool

class aries_cloudagent.vc.ld_proofs.**KeyPair**

Bases: abc.ABC

Base key pair class.

abstract from_verification_method(*verification_method: dict*) → *aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair*

Create new key pair class based on the passed verification method.

abstract property has_public_key: bool

Whether key pair has a public key.

Public key is required for verification, but can be set dynamically in the verification process.

abstract property public_key: Optional[bytes]

Getter for the public key bytes.

Returns The public key

Return type bytes

abstract async sign(*message: Union[List[bytes], bytes]*) → bytes
Sign message(s) using key pair.

abstract async verify(*message: Union[List[bytes], bytes]*, *signature: bytes*) → bool
Verify message(s) against signature using key pair.

class aries_cloudagent.vc.ld_proofs.**LinkedDataProof**(*, *signature_type: str*, *proof: Optional[dict] = None*, *supported_derive_proof_types: Optional[List[str]] = None*)

Bases: abc.ABC

Base Linked data proof.

async create_proof(*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → dict
Create proof for document.

Parameters

- **document** (*dict*) – The document to create the proof for
- **purpose** (ProofPurpose) – The proof purpose to include in the proof
- **document_loader** (DocumentLoader) – Document loader used for resolving

Returns The proof object

Return type dict

async derive_proof(*, *proof: dict*, *document: dict*, *reveal_document: dict*, *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*) → *aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.DeriveProofResult*
Derive proof for document, returning derived document and proof.

Parameters

- **proof** (*dict*) – The proof to derive from
- **document** (*dict*) – The document to derive the proof for
- **reveal_document** (*dict*) – The JSON-LD frame the revealed attributes
- **document_loader** (DocumentLoader) – Document loader used for resolving
- **nonce** (*bytes, optional*) – Nonce to use for the proof. Defaults to None.

Returns The derived document and proof

Return type DeriveProofResult

match_proof(*signature_type: str*) → bool
Match signature type to signature type of this suite.

async verify_proof(*, *proof: dict*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*
Verify proof against document and proof purpose.

Parameters

- **proof** (*dict*) – The proof to verify
- **document** (*dict*) – The document to verify the proof against
- **purpose** (ProofPurpose) – The proof purpose to verify the proof against
- **document_loader** (DocumentLoader) – Document loader used for resolving

Returns The results of the proof verification

Return type ValidationResult

exception aries_cloudagent.vc.ld_proofs.**LinkedDataProofException**

Bases: Exception

Base exception for linked data proof module.

class aries_cloudagent.vc.ld_proofs.**LinkedDataSignature**(*, *signature_type: str*, *proof: Optional[dict] = None*, *verification_method: Optional[str] = None*, *date: Optional[datetime.datetime] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof*

Linked Data Signature class.

async create_proof(*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → dict

Create proof for document, return proof.

abstract async sign(*, *verify_data: bytes*, *proof: dict*) → dict

Sign the data and add it to the proof.

Parameters

- **verify_data** (*bytes*) – The data to sign.
- **proof** (*dict*) – The proof to add the signature to

Returns The proof object with the added signature

Return type dict

async verify_proof(*, *proof: dict*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*

Verify proof against document and proof purpose.

abstract async verify_signature(*, *verify_data: bytes*, *verification_method: dict*, *document: dict*, *proof: dict*, *document_loader: Callable[[str, dict], dict]*) → bool

Verify the data against the proof.

Parameters

- **verify_data** (*bytes*) – The data to check
- **verification_method** (*dict*) – The verification method to use.
- **document** (*dict*) – The document the verify data is derived for as extra context
- **proof** (*dict*) – The proof to check
- **document_loader** (DocumentLoader) – Document loader used for resolving

Returns Whether the signature is valid for the data

Return type bool

class aries_cloudagent.vc.ld_proofs.**ProofPurpose**(*, *term: str*, *date: Optional[datetime.datetime] = None*, *max_timestamp_delta: Optional[datetime.timedelta] = None*)

Bases: object

Base proof purpose class.

match(*proof: dict*) → bool
Check whether the passed proof matches with the term of this proof purpose.

update(*proof: dict*) → dict
Update proof purpose on proof.

validate(*, *proof: dict*, *document: dict*, *suite:* LinkedDataProof, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*
Validate whether created date of proof is out of max_timestamp_delta range.

class aries_cloudagent.vc.ld_proofs.**ProofResult**(*, *verified: bool*, *proof: Optional[dict] = None*, *error: Optional[Exception] = None*, *purpose_result: Optional[*aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*] = None*)

Bases: object

Proof result class.

class aries_cloudagent.vc.ld_proofs.**ProofSet**

Bases: object

Class for managing proof sets on a JSON-LD document.

async static **add**(*, *document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → dict
Add a Linked Data proof to the document.

If the document contains other proofs, the proof will be appended to the existing set of proofs.

Important note: This method assumes that the term *proof* in the given document has the same definition as the *https://w3id.org/security/v2* JSON-LD @context.

Parameters

- **document** (*dict*) – JSON-LD document to be signed.
- **suite** (LinkedDataProof) – A signature suite instance that will create the proof
- **purpose** (ProofPurpose) – A proof purpose instance that will augment the proof with information describing its intended purpose.
- **document_loader** (DocumentLoader) – Document loader to use.

Returns

The signed document, with the signature in the top-level *proof* property.

Return type dict

async static derive(*, *document: dict*, *reveal_document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof, *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*) → dict

Create new derived Linked Data proof(s) on document using the reveal document.

Important note: This method assumes that the term *proof* in the given document has the same definition as the *https://w3id.org/security/v2* JSON-LD @context. (v3 because BBS?)

Parameters

- **document** (*dict*) – JSON-LD document with one or more proofs to be derived.
- **reveal_document** (*dict*) – JSON-LD frame specifying the attributes to reveal.
- **suite** (LinkedDataProof) – A signature suite instance to derive the proof.
- **document_loader** (DocumentLoader) – Document loader to use.
- **nonce** (*bytes, optional*) – Nonce to use for the proof. Defaults to None.

Returns

The derived document with the derived proof(s) in the top-level *proof* property.

Return type dict

async static verify(*, *document: dict*, *suites: List[*aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof*]*, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.DocumentVerificationResult*

Verify Linked Data proof(s) on a document.

The proofs to be verified must match the given proof purse.

Important note: This method assumes that the term *proof* in the given document has the same definition as the *https://w3id.org/security/v2* JSON-LD @context.

Parameters

- **document** (*dict*) – JSON-LD document with one or more proofs to be verified.
- **suites** (*List[*LinkedDataProof*]*) – Acceptable signature suite instances for verifying the proof(s).
- **purpose** (ProofPurpose) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose.
- **document_loader** (DocumentLoader) – Document loader to use.

Returns

Object with a *verified* property that is *true* if at least one proof matching the given purpose and suite verifies and *false* otherwise. Also contains *errors* and *results* properties with extra data.

Return type *DocumentVerificationResult*

class aries_cloudagent.vc.ld_proofs.**PurposeResult**(*, *valid: bool*, *error: Optional[Exception] = None*, *controller: Optional[dict] = None*)

Bases: object

Proof purpose result class.

class aries_cloudagent.vc.ld_proofs.**WalletKeyPair**(*, *wallet:* aries_cloudagent.wallet.base.BaseWallet, *key_type:* aries_cloudagent.wallet.key_type.KeyType, *public_key_base58: Optional[str] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair*

Base wallet key pair.

from_verification_method(*verification_method: dict*) → *aries_cloudagent.vc.ld_proofs.crypto.wallet_key_pair.WalletKeyPair*

Create new WalletKeyPair from public key in verification method.

property has_public_key: bool

Whether key pair has public key.

property public_key: Optional[bytes]

Getter for public key.

async sign(*message: Union[List[bytes], bytes]*) → bytes

Sign message using wallet.

async verify(*message: Union[List[bytes], bytes]*, *signature: bytes*) → bool

Verify message against signature using wallet.

async aries_cloudagent.vc.ld_proofs.**derive**(*, *document: dict*, *reveal_document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof, *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*) → dict

Derive proof(s) for document with reveal document.

All proofs matching the signature suite type will be replaced with a derived proof. Other proofs will be discarded.

Parameters

- **document** (*dict*) – The document with one or more proofs to be derived
- **reveal_document** (*dict*) – The JSON-LD frame specifying the revealed attributes
- **suite** (LinkedDataProof) – The linked data signature cryptographic suite with which to derive the proof
- **document_loader** (DocumentLoader) – The document loader to use.
- **nonce** (*bytes, optional*) – Nonce to use for the proof. Defaults to None.

Returns The document with derived proof(s).

Return type dict

aries_cloudagent.vc.ld_proofs.**get_properties_without_context**(*document: dict*, *document_loader: Callable[[str, dict], dict]*) → Sequence[str]

Get the properties from document that don't have an context definition.

async aries_cloudagent.vc.ld_proofs.**sign**(*, *document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → dict

Cryptographically signs the provided document by adding a *proof* section.

Proof is added based on the provided suite and proof purpose

Parameters

- **document** (*dict*) – JSON-LD document to be signed.
- **suite** (LinkedDataProof) – The linked data signature cryptographic suite with which to sign the document
- **purpose** (ProofPurpose) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose.
- **document_loader** (DocumentLoader) – The document loader to use.

Raises *LinkedDataProofException* – When a jsonld url cannot be resolved, OR signing fails.

Returns Signed document.

Return type dict

async aries_cloudagent.vc.ld_proofs.**verify**(*, *document: dict*, *suites: List[aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof]*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.DocumentVerificationResult*

Verify the linked data signature on the provided document.

Parameters

- **document** (*dict*) – The document with one or more proofs to be verified.
- **suites** (*List[LinkedDataProof]*) – Acceptable signature suite instances for verifying the proof(s).
- **purpose** (ProofPurpose) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose.
- **document_loader** (DocumentLoader) – The document loader to use.

Returns

Object with a *verified* boolean property that is *True* if at least one proof matching the given purpose and suite verifies and *False* otherwise. a *results* property with an array of detailed results. if *False* an *errors* property will be present, with a list containing all of the errors that occurred during the verification process.

Return type *DocumentVerificationResult*

Subpackages

aries_cloudagent.vc.ld_proofs.crypto package

Submodules

aries_cloudagent.vc.ld_proofs.crypto.key_pair module

Base key pair class.

class aries_cloudagent.vc.ld_proofs.crypto.key_pair.**KeyPair**

Bases: abc.ABC

Base key pair class.

abstract from_verification_method(*verification_method: dict*) → *aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair*
Create new key pair class based on the passed verification method.

abstract property has_public_key: bool
Whether key pair has a public key.

Public key is required for verification, but can be set dynamically in the verification process.

abstract property public_key: Optional[bytes]
Getter for the public key bytes.

Returns The public key

Return type bytes

abstract async sign(*message: Union[List[bytes], bytes]*) → bytes
Sign message(s) using key pair.

abstract async verify(*message: Union[List[bytes], bytes]*, *signature: bytes*) → bool
Verify message(s) against signature using key pair.

aries_cloudagent.vc.ld_proofs.crypto.wallet_key_pair module

Key pair based on base wallet interface.

class aries_cloudagent.vc.ld_proofs.crypto.wallet_key_pair.**WalletKeyPair**(*, *wallet:* aries_cloudagent.wallet.base.BaseWalle *key_type:* aries_cloudagent.wallet.key_type.KeyTy *public_key_base58: Optional[str] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.crypto.key_pair.KeyPair*

Base wallet key pair.

from_verification_method(*verification_method: dict*) → *aries_cloudagent.vc.ld_proofs.crypto.wallet_key_pair.WalletKeyPair*
Create new WalletKeyPair from public key in verification method.

property has_public_key: bool
Whether key pair has public key.

property public_key: Optional[bytes]
Getter for public key.

async sign(*message: Union[List[bytes], bytes]*) → bytes
Sign message using wallet.

async verify(*message: Union[List[bytes], bytes]*, *signature: bytes*) → bool
Verify message against signature using wallet.

aries_cloudagent.vc.ld_proofs.purposes package

Submodules

aries_cloudagent.vc.ld_proofs.purposes.assertion_proof_purpose module

Assertion proof purpose class.

class aries_cloudagent.vc.ld_proofs.purposes.assertion_proof_purpose.**AssertionProofPurpose**(*, *date: Optional[datetime.* = *None*, *max_timestamp_* *Optional[datetime.* = *None*)

Bases: *aries_cloudagent.vc.ld_proofs.purposes.controller_proof_purpose.ControllerProofPurpose*

Assertion proof purpose class.

term = 'assertionMethod'

aries_cloudagent.vc.ld_proofs.purposes.authentication_proof_purpose module

Authentication proof purpose class.

class aries_cloudagent.vc.ld_proofs.purposes.authentication_proof_purpose.**AuthenticationProofPurpose**(*, *cha* *len* *str*, *do-* *ma* *Op* *tio* = *No* *dat* *Op* *tio* = *No* *ma* *Op* *tio* = *No*

Bases: *aries_cloudagent.vc.ld_proofs.purposes.controller_proof_purpose.ControllerProofPurpose*

Authentication proof purpose.

term = 'authentication'

update(*proof: dict*) → dict
Update poof purpose, challenge and domain on proof.

validate(*, *proof: dict*, *document: dict*, *suite: LinkedDataProof*, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*
Validate whether challenge and domain are valid.

aries_cloudagent.vc.ld_proofs.purposes.controller_proof_purpose module

Controller proof purpose class.

class aries_cloudagent.vc.ld_proofs.purposes.controller_proof_purpose.**ControllerProofPurpose**(*, *term: str*, *date: Optional[datetim = None*, *max_timestan Optional[datetim = None*)

Bases: *aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*

Controller proof purpose class.

validate(*, *proof: dict*, *document: dict*, *suite: LinkedDataProof*, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*
Validate whether verification method of proof is authorized by controller.

aries_cloudagent.vc.ld_proofs.purposes.credential_issuance_purpose module

Credential Issuance proof purpose class.

class aries_cloudagent.vc.ld_proofs.purposes.credential_issuance_purpose.**CredentialIssuancePurpose**(*, *date: Optional = None*, *max_ Optional = None*)

Bases: *aries_cloudagent.vc.ld_proofs.purposes.assertion_proof_purpose.AssertionProofPurpose*

Credential Issuance proof purpose.

validate(*, *proof: dict*, *document: dict*, *suite: LinkedDataProof*, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*

Validate if the issuer matches the controller of the verification method.

aries_cloudagent.vc.ld_proofs.purposes.proof_purpose module

Base Proof Purpose class.

class aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.**ProofPurpose**(*, *term: str*, *date: Optional[datetime.datetime] = None*, *max_timestamp_delta: Optional[datetime.timedelta] = None*)

Bases: object

Base proof purpose class.

match(*proof: dict*) → bool

Check whether the passed proof matches with the term of this proof purpose.

update(*proof: dict*) → dict

Update proof purpose on proof.

validate(*, *proof: dict*, *document: dict*, *suite: LinkedDataProof*, *verification_method: dict*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.PurposeResult*

Validate whether created date of proof is out of max_timestamp_delta range.

aries_cloudagent.vc.ld_proofs.suites package

Submodules

aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020 module

BbsBlsSignature2020 class.

class aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020.**BbsBlsSignature2020**(*, *key_pair: aries_cloudagent.vc.ld_proofs.key_pair.KeyPair*, *proof: Optional[dict] = None*, *verification_method: Optional[str] = None*, *date: Optional[datetime.datetime] = None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020_base.BbsBlsSignature2020Base*

BbsBlsSignature2020 class.

async create_proof(*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → dict

Create proof for document, return proof.

async sign(*, *verify_data: List[bytes]*, *proof: dict*) → dict

Sign the data and add it to the proof.

Parameters

- **verify_data** (*List[bytes]*) – The data to sign.
- **proof** (*dict*) – The proof to add the signature to

Returns The proof object with the added signature

Return type dict

signature_type = 'BbsBlsSignature2020'

async verify_proof(*, *proof: dict*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*

Verify proof against document and proof purpose.

async verify_signature(*, *verify_data: List[bytes]*, *verification_method: dict*, *document: dict*, *proof: dict*, *document_loader: Callable[[str, dict], dict]*) → bool

Verify the data against the proof.

Parameters

- **verify_data** (*bytes*) – The data to check
- **verification_method** (*dict*) – The verification method to use.

- **document** (*dict*) – The document the verify data is derived for as extra context
- **proof** (*dict*) – The proof to check
- **document_loader** (*DocumentLoader*) – Document loader used for resolving

Returns Whether the signature is valid for the data

Return type bool

aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020_base module

BbsBlsSignature2020Base class.

class aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020_base.**BbsBlsSignature2020Base**(*, *signature_type: str*, *proof: Optional[dict* = *None*, *supported_der Optional[List* = *None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof*

Base class for BbsBlsSignature suites.

BBS_SUPPORTED = False

aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_proof_2020 module

BbsBlsSignatureProof2020 class.

class aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_proof_2020.**BbsBlsSignatureProof2020**(*, *key_pai* aries_cl

Bases: *aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020_base.BbsBlsSignature2020Base*

BbsBlsSignatureProof2020 class.

async derive_proof(*, *proof: dict*, *document: dict*, *reveal_document: dict*, *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*)

Derive proof for document, return dict with derived document and proof.

signature_type = 'BbsBlsSignatureProof2020'

supported_derive_proof_types = ['BbsBlsSignature2020', 'sec:BbsBlsSignature2020', 'https://w3id.org/security#BbsBlsSignature2020']

async verify_proof(*, *proof: dict*, *document: dict*, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*

Verify proof against document and proof purpose.

aries_cloudagent.vc.ld_proofs.suites.ed25519_signature_2018 module

Ed25519Signature2018 suite.

class aries_cloudagent.vc.ld_proofs.suites.ed25519_signature_2018.**Ed25519Signature2018**(*, *key_pair:* aries_cloudagent.vc.l *proof: Optional[dict]* = *None*, *verification_method: Optional[str]* = *None*, *date: Optional[Union[datetim str]]* = *None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.jws_linked_data_signature.JwsLinkedDataSignature*

Ed25519Signature2018 suite.

signature_type = 'Ed25519Signature2018'

aries_cloudagent.vc.ld_proofs.suites.jws_linked_data_signature module

JWS Linked Data class.

class aries_cloudagent.vc.ld_proofs.suites.jws_linked_data_signature.**JwsLinkedDataSignature**(*, *signature_type: str*, *algorithm: str*, *required_key_typ str*, *key_pair: aries_cloudage proof: Optional[dict]* = *None*, *verification_method: Optional[str]* = *None*, *date: Optional[Union[d str]]* = *None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.linked_data_signature.LinkedDataSignature*

JWS Linked Data class.

async sign(*, *verify_data: bytes*, *proof: dict*) → dict

Sign the data and add it to the proof.

Adds a jws to the proof that can be used for multiple signature algorithms.

Parameters

- **verify_data** (*bytes*) – The data to sign.
- **proof** (*dict*) – The proof to add the signature to

Returns The proof object with the added signature

Return type dict

async verify_signature(*, *verify_data: bytes*, *verification_method: dict*, *document: dict*, *proof: dict*, *document_loader: Callable[[str, dict], dict]*)

Verify the data against the proof.

Checks for a jws on the proof.

Parameters

- **verify_data** (*bytes*) – The data to check
- **verification_method** (*dict*) – The verification method to use.
- **document** (*dict*) – The document the verify data is derived for as extra context
- **proof** (*dict*) – The proof to check
- **document_loader** (*DocumentLoader*) – Document loader used for resolving

Returns Whether the signature is valid for the data

Return type bool

aries_cloudagent.vc.ld_proofs.suites.linked_data_proof module

Abstract base class for linked data proofs.

class aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.**DeriveProofResult**(**args: Any*, ***kwargs: Any*)

Bases: typing_extensions.

Result dict for deriving a proof.

document

proof

class aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.**LinkedDataProof**(*, *signature_type: str*, *proof: Optional[dict] = None*, *supported_derive_proof_types: Optional[List[str]] = None*)

Bases: abc.ABC

Base Linked data proof.

async create_proof(*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → dict

Create proof for document.

Parameters

- **document** (*dict*) – The document to create the proof for
- **purpose** (*ProofPurpose*) – The proof purpose to include in the proof
- **document_loader** (*DocumentLoader*) – Document loader used for resolving

Returns The proof object

Return type dict

async derive_proof(*, *proof: dict*, *document: dict*, *reveal_document: dict*, *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*) → *aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.DeriveProofResult*

Derive proof for document, returning derived document and proof.

Parameters

- **proof** (*dict*) – The proof to derive from
- **document** (*dict*) – The document to derive the proof for
- **reveal_document** (*dict*) – The JSON-LD frame the revealed attributes
- **document_loader** (*DocumentLoader*) – Document loader used for resolving
- **nonce** (*bytes, optional*) – Nonce to use for the proof. Defaults to None.

Returns The derived document and proof

Return type *DeriveProofResult*

match_proof(*signature_type: str*) → bool

Match signature type to signature type of this suite.

async verify_proof(*, *proof: dict*, *document: dict*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*

Verify proof against document and proof purpose.

Parameters

- **proof** (*dict*) – The proof to verify
- **document** (*dict*) – The document to verify the proof against
- **purpose** (*ProofPurpose*) – The proof purpose to verify the proof against
- **document_loader** (*DocumentLoader*) – Document loader used for resolving

Returns The results of the proof verification

Return type ValidationResult

aries_cloudagent.vc.ld_proofs.suites.linked_data_signature module

Linked Data Signature class.

class aries_cloudagent.vc.ld_proofs.suites.linked_data_signature.**LinkedDataSignature**(*, *signature_type: str*, *proof: Optional[dict]* = *None*, *verification_method: Optional[str]* = *None*, *date: Optional[datetime.datetime* = *None*)

Bases: *aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof*

Linked Data Signature class.

async create_proof(*, *document: dict*, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → dict

Create proof for document, return proof.

abstract async sign(*, *verify_data: bytes*, *proof: dict*) → dict

Sign the data and add it to the proof.

Parameters

- **verify_data** (*bytes*) – The data to sign.
- **proof** (*dict*) – The proof to add the signature to

Returns The proof object with the added signature

Return type dict

async verify_proof(*, *proof: dict*, *document: dict*, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.ProofResult*

Verify proof against document and proof purpose.

abstract async verify_signature(*, *verify_data: bytes*, *verification_method: dict*, *document: dict*, *proof: dict*, *document_loader: Callable[[str, dict], dict]*) → bool

Verify the data against the proof.

Parameters

- **verify_data** (*bytes*) – The data to check
- **verification_method** (*dict*) – The verification method to use.
- **document** (*dict*) – The document the verify data is derived for as extra context

- **proof** (*dict*) – The proof to check
- **document_loader** (*DocumentLoader*) – Document loader used for resolving

Returns Whether the signature is valid for the data

Return type bool

Submodules

aries_cloudagent.vc.ld_proofs.check module

Validator methods to check for properties without a context.

aries_cloudagent.vc.ld_proofs.check.**diff_dict_keys**(*full: dict*, *with_missing: dict*, *prefix: Optional[str] = None*, ***, *document_loader: Callable[[str, dict], dict]*, *context*) → Sequence[str]

Get the difference in dict keys between full and with_missing.

Checks recursively

Parameters

- **full** (*dict*) – The full dict with all keys present
- **with_missing** (*dict*) – The dict with possibly keys missing
- **prefix** (*str, optional*) – The prefix. Mostly used for internal recursion.

Returns List of missing property names in with_missing

Return type Sequence[str]

aries_cloudagent.vc.ld_proofs.check.**get_properties_without_context**(*document: dict*, *document_loader: Callable[[str, dict], dict]*) → Sequence[str]

Get the properties from document that don't have an context definition.

aries_cloudagent.vc.ld_proofs.constants module

JSON-LD, Linked Data Proof and Verifiable Credential constants.

aries_cloudagent.vc.ld_proofs.document_loader module

JSON-LD document loader methods.

class aries_cloudagent.vc.ld_proofs.document_loader.**DocumentLoader**(*profile: aries_cloudagent.core.profile.Profile*, *cache_ttl: int = 300*)

Bases: object

JSON-LD document loader.

async load_document(*url: str*, *options: dict*)

Load JSON-LD document.

Method signature conforms to PyLD document loader interface

Document loading is processed in separate thread to deal with async to sync transformation.

aries_cloudagent.vc.ld_proofs.error module

Linked data proof exception classes.

exception aries_cloudagent.vc.ld_proofs.error.**LinkedDataProofException**
Bases: Exception

Base exception for linked data proof module.

aries_cloudagent.vc.ld_proofs.ld_proofs module

Linked data proof signing and verification methods.

async aries_cloudagent.vc.ld_proofs.ld_proofs.**derive**(*, *document: dict*, *reveal_document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDa *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*) → dict

Derive proof(s) for document with reveal document.

All proofs matching the signature suite type will be replaced with a derived proof. Other proofs will be discarded.

Parameters

- **document** (*dict*) – The document with one or more proofs to be derived
- **reveal_document** (*dict*) – The JSON-LD frame specifying the revealed attributes
- **suite** (*LinkedDataProof*) – The linked data signature cryptographic suite with which to derive the proof
- **document_loader** (*DocumentLoader*) – The document loader to use.
- **nonce** (*bytes, optional*) – Nonce to use for the proof. Defaults to None.

Returns The document with derived proof(s).

Return type dict

async aries_cloudagent.vc.ld_proofs.ld_proofs.**sign**(*, *document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataP *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose *document_loader: Callable[[str, dict], dict]*) → dict

Cryptographically signs the provided document by adding a *proof* section.

Proof is added based on the provided suite and proof purpose

Parameters

- **document** (*dict*) – JSON-LD document to be signed.
- **suite** (*LinkedDataProof*) – The linked data signature cryptographic suite with which to sign the document
- **purpose** (*ProofPurpose*) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose.
- **document_loader** (*DocumentLoader*) – The document loader to use.

Raises **LinkedDataProofException** – When a jsonld url cannot be resolved, OR signing fails.

Returns Signed document.

Return type dict

async aries_cloudagent.vc.ld_proofs.ld_proofs.**verify**(*, *document: dict*, *suites: List[aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.Linke* *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurp* *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.DocumentVerifica*

Verify the linked data signature on the provided document.

Parameters

- **document** (*dict*) – The document with one or more proofs to be verified.
- **suites** (*List[LinkedDataProof]*) – Acceptable signature suite instances for verifying the proof(s).
- **purpose** (*ProofPurpose*) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose.
- **document_loader** (*DocumentLoader*) – The document loader to use.

Returns

Object with a *verified* boolean property that is *True* if at least one proof matching the given purpose and suite verifies and *False* otherwise. a *results* property with an array of detailed results. if *False* an *errors* property will be present, with a list containing all of the errors that occurred during the verification process.

Return type DocumentVerificationResult

aries_cloudagent.vc.ld_proofs.proof_set module

Class to represent a Linked Data proof set.

class aries_cloudagent.vc.ld_proofs.proof_set.**ProofSet**

Bases: object

Class for managing proof sets on a JSON-LD document.

async static **add**(*, *document: dict*, *suite: aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof*, *purpose: aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*, *document_loader: Callable[[str, dict], dict]*) → dict

Add a Linked Data proof to the document.

If the document contains other proofs, the proof will be appended to the existing set of proofs.

Important note: This method assumes that the term *proof* in the given document has the same definition as the *https://w3id.org/security/v2* JSON-LD @context.

Parameters

- **document** (*dict*) – JSON-LD document to be signed.
- **suite** (*LinkedDataProof*) – A signature suite instance that will create the proof

- **purpose** (*ProofPurpose*) – A proof purpose instance that will augment the proof with information describing its intended purpose.
- **document_loader** (*DocumentLoader*) – Document loader to use.

Returns

The signed document, with the signature in the top-level *proof* property.

Return type dict

async static derive(*, *document: dict*, *reveal_document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof, *document_loader: Callable[[str, dict], dict]*, *nonce: Optional[bytes] = None*) → dict

Create new derived Linked Data proof(s) on document using the reveal document.

Important note: This method assumes that the term *proof* in the given document has the same definition as the *https://w3id.org/security/v2* JSON-LD @context. (v3 because BBS?)

Parameters

- **document** (*dict*) – JSON-LD document with one or more proofs to be derived.
- **reveal_document** (*dict*) – JSON-LD frame specifying the attributes to reveal.
- **suite** (*LinkedDataProof*) – A signature suite instance to derive the proof.
- **document_loader** (*DocumentLoader*) – Document loader to use.
- **nonce** (*bytes, optional*) – Nonce to use for the proof. Defaults to None.

Returns

The derived document with the derived proof(s) in the top-level *proof* property.

Return type dict

async static verify(*, *document: dict*, *suites:* *List[*aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof*]*, *purpose:* aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose, *document_loader: Callable[[str, dict], dict]*) → *aries_cloudagent.vc.ld_proofs.validation_result.DocumentVerificationResult*

Verify Linked Data proof(s) on a document.

The proofs to be verified must match the given proof purse.

Important note: This method assumes that the term *proof* in the given document has the same definition as the *https://w3id.org/security/v2* JSON-LD @context.

Parameters

- **document** (*dict*) – JSON-LD document with one or more proofs to be verified.
- **suites** (*List[LinkedDataProof]*) – Acceptable signature suite instances for verifying the proof(s).
- **purpose** (*ProofPurpose*) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose.
- **document_loader** (*DocumentLoader*) – Document loader to use.

Returns

Object with a *verified* property that is *true* if at least one proof matching the given purpose and suite verifies and *false* otherwise. Also contains *errors* and *results* properties with extra data.

Return type DocumentVerificationResult

aries_cloudagent.vc.ld_proofs.validation_result module

Proof verification and validation result classes.

class aries_cloudagent.vc.ld_proofs.validation_result.**DocumentVerificationResult**(*, *verified: bool*, *document: Optional[dict] = None*, *results: Optional[List[aries_cloudagent. = None*, *errors: Optional[List[Exception]] = None*)

Bases: object

Domain verification result class.

class aries_cloudagent.vc.ld_proofs.validation_result.**ProofResult**(*, *verified: bool*, *proof: Optional[dict] = None*, *error: Optional[Exception] = None*, *purpose_result: Optional[aries_cloudagent.vc.ld_proofs.validation_ = None*)

Bases: object

Proof result class.

class aries_cloudagent.vc.ld_proofs.validation_result.**PurposeResult**(*, *valid: bool*, *error: Optional[Exception] = None*, *controller: Optional[dict] = None*)

Bases: object

Proof purpose result class.

aries_cloudagent.vc.vc_ld package

class aries_cloudagent.vc.vc_ld.**CredentialSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Linked data credential schema.

Based on https://www.w3.org/TR/vc-data-model

class Meta
Bases: object

Accept parameter overload.

model_class
alias of *aries_cloudagent.vc.vc_ld.models.credential.VerifiableCredential*

add_unknown_properties(*data: dict*, *original*, ***kwargs*)
Add back unknown properties before outputting.

context

credential_subject
Dict or Dict List field for Marshmallow.

expiration_date

id

issuance_date

issuer
URI or Dict field for Marshmallow.

proof

type

class aries_cloudagent.vc.vc_ld.**LDProof**(*type: Optional[str] = None*, *proof_purpose: Optional[str] = None*, *verification_method: Optional[str] = None*, *created: Optional[str] = None*, *domain: Optional[str] = None*, *challenge: Optional[str] = None*, *jws: Optional[str] = None*, *proof_value: Optional[str] = None*, *nonce: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.messaging.models.base.BaseModel*

Linked Data Proof model.

class Meta
Bases: object

LinkedDataProof metadata.

schema_class = 'LinkedDataProofSchema'

class aries_cloudagent.vc.vc_ld.**LinkedDataProofSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Linked data proof schema.

Based on https://w3c-ccg.github.io/ld-proofs

class Meta
Bases: object

Accept parameter overload.

model_class
alias of *aries_cloudagent.vc.vc_ld.models.linked_data_proof.LDProof*

add_unknown_properties(*data: dict*, *original*, ***kwargs*)
Add back unknown properties before outputting.

challenge

created

domain

jws

nonce

proof_purpose

proof_value

type

verification_method

class aries_cloudagent.vc.vc_ld.**PresentationVerificationResult**(***, *verified: bool*, *presentation_result: Optional[*aries_cloudagent.vc.ld_proofs.validation_resu *= None*, *credential_results: Optional[List[*aries_cloudagent.vc.ld_proofs.validation_ *= None*, *errors: Optional[List[Exception]] = None*)

Bases: object

Presentation verification result class.

class aries_cloudagent.vc.vc_ld.**VerifiableCredential**(*context: Optional[List[Union[str, dict]]] = None*, *id: Optional[str] = None*, *type: Optional[List[str]] = None*, *issuer: Optional[Union[dict, str]] = None*, *issuance_date: Optional[str] = None*, *expiration_date: Optional[str] = None*, *credential_subject: Optional[Union[dict, List[dict]]] = None*, *proof: Optional[Union[dict,* aries_cloudagent.vc.vc_ld.models.linked_data_proof.LDProof*]] = None*, ***kwargs*)

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

Verifiable Credential model.

class Meta
Bases: object

VerifiableCredential metadata.

schema_class = 'CredentialSchema'

add_context(*context: Union[str, dict]*)
Add a context to this credential.

add_type(*type: str*)
Add a type to this credential.

property context
Getter for context.

property context_urls: List[str]
Getter for context urls.

property credential_subject
Getter for credential subject.

property credential_subject_ids: List[str]
Getter for credential subject ids.

property expiration_date
Getter for expiration date.

property id
Getter for id.

property issuance_date
Getter for issuance date.

property issuer
Getter for issuer.

property issuer_id: Optional[str]
Getter for issuer id.

property proof
Getter for proof.

property type: List[str]
Getter for type.

class aries_cloudagent.vc.vc_ld.**VerifiableCredentialSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Linked data verifiable credential schema.

Based on https://www.w3.org/TR/vc-data-model

proof

async aries_cloudagent.vc.vc_ld.**create_presentation**(*, *credentials: List[dict]*, *presentation_id: Optional[str] = None*) → dict
Create presentation and add the credentials to it.

Will validates the structure off all credentials, but does not sign the presentation yet. Call sing_presentation to do this.

Parameters

- **credentials** (*List[dict]*) – Credentails to add to the presentation
- **presentation_id** (*str, optional*) – Id of the presentation. Defaults to None.

Raises **LinkedDataProofException** – When not all credentials have a valid structure

Returns The unsigned presentation object

Return type dict

async aries_cloudagent.vc.vc_ld.**derive_credential**(*, *credential: dict*, *reveal_document: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataPr *document_loader: Callable[[str, dict], dict]*) → dict

Derive new credential from the existing credential and the reveal document.

All proofs matching the signature suite type will be replaced with a derived proof. Other proofs will be discarded.

Parameters

- **credential** (*dict*) – The credential to derive the new credential from.
- **reveal_document** (*dict*) – JSON-LD frame to select which attributes to include.
- **suite** (*LinkedDataProof*) – The signature suite to use for derivation
- **document_loader** (*DocumentLoader*) – The document loader to use.

Returns The derived credential.

Return type dict

async aries_cloudagent.vc.vc_ld.**issue_vc**(*, *credential: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof, *document_loader: Callable[[str, dict], dict]*, *purpose: Optional[*aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*]* = *None*) → dict

Issue a verifiable credential.

Takes the base credentail document, verifies it, and adds a digital signature to it.

Parameters

- **credential** (*dict*) – Base credential document.
- **suite** (*LinkedDataProof*) – Signature suite to sign the credential with.
- **document_loader** (*DocumentLoader*) – Document loader to use
- **purpose** (*ProofPurpose, optional*) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose. Default to CredentialIssuancePurpose

Raises **LinkedDataProofException** – When the credential has an invalid structure OR signing fails

Returns The signed verifiable credential

Return type dict

async aries_cloudagent.vc.vc_ld.**sign_presentation**(*, *presentation: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataPr *document_loader: Callable[[str, dict], dict]*, *purpose: Optional[*aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPu = *None*, *challenge: Optional[str]* = *None*, *domain: Optional[str]* = *None*) → dict

Sign the presentation with the passed signature suite.

Will set a default AuthenticationProofPurpose if no proof purpose is passed.

Parameters

- **presentation** (*dict*) – The presentation to sign

- **suite** (*LinkedDataProof*) – The signature suite to sign the presentation with
- **document_loader** (*DocumentLoader*) – Document loader to use.
- **purpose** (*ProofPurpose, optional*) – Purpose to use. Required if challenge is None
- **challenge** (*str, optional*) – Challenge to use. Required if domain is None.
- **domain** (*str, optional*) – Domain to use. Only used if purpose is None.

Raises **LinkedDataProofException** – When both purpose and challenge are not provided And when signing of the presentation fails

Returns A verifiable presentation object

Return type dict

async aries_cloudagent.vc.vc_ld.**verify_credential**(*, *credential: dict*, *suites: List[aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDa document_loader: Callable[[str, dict], dict], purpose: Optional[aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPu = None*) → *aries_cloudagent.vc.ld_proofs.validation_result.DocumentVerification*

Verify credential structure, proof purpose and signature.

Parameters

- **credential** (*dict*) – The credential to verify
- **suites** (*List[LinkedDataProof]*) – The signature suites to verify with
- **document_loader** (*DocumentLoader*) – Document loader used for resolving of documents
- **purpose** (*ProofPurpose, optional*) – Proof purpose to use. Defaults to CredentialIssuancePurpose

Returns

The result of the verification. Verified property indicates whether the verification was successful

Return type DocumentVerificationResult

async aries_cloudagent.vc.vc_ld.**verify_presentation**(*, *presentation: dict*, *suites: List[aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.Linked document_loader: Callable[[str, dict], dict], purpose: Optional[aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.Proo = None*, *challenge: Optional[str] = None*, *domain: Optional[str] = None*) → *aries_cloudagent.vc.vc_ld.validation_result.PresentationVerificatio*

Verify presentation structure, credentials, proof purpose and signature.

Parameters

- **presentation** (*dict*) – The presentation to verify
- **suites** (*List[LinkedDataProof]*) – The signature suites to verify with
- **document_loader** (*DocumentLoader*) – Document loader used for resolving of documents
- **purpose** (*ProofPurpose, optional*) – Proof purpose to use. Defaults to AuthenticationProofPurpose

- **challenge** (*str, optional*) – The challenge to use for authentication. Required if purpose is not passed, not used if purpose is passed
- **domain** (*str, optional*) – Domain to use for the authentication proof purpose. Not used if purpose is passed

Returns

The result of the verification. Verified property indicates whether the verification was successful

Return type *PresentationVerificationResult*

Subpackages

aries_cloudagent.vc.vc_ld.models package

Submodules

aries_cloudagent.vc.vc_ld.models.credential module

Verifiable Credential marshmallow schema classes.

class aries_cloudagent.vc.vc_ld.models.credential.**CredentialSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Linked data credential schema.

Based on https://www.w3.org/TR/vc-data-model

class Meta

Bases: object

Accept parameter overload.

model_class

alias of *aries_cloudagent.vc.vc_ld.models.credential.VerifiableCredential*

add_unknown_properties(*data: dict*, *original*, ***kwargs*)

Add back unknown properties before outputting.

context

credential_subject

Dict or Dict List field for Marshmallow.

expiration_date

id

issuance_date

issuer

URI or Dict field for Marshmallow.

proof

type

class aries_cloudagent.vc.vc_ld.models.credential.**VerifiableCredential**(*context: Optional[List[Union[str, dict]]] = None*, *id: Optional[str] = None*, *type: Optional[List[str]] = None*, *issuer: Optional[Union[dict, str]] = None*, *issuance_date: Optional[str] = None*, *expiration_date: Optional[str] = None*, *credential_subject: Optional[Union[dict, List[dict]]] = None*, *proof: Optional[Union[dict, aries_cloudagent.vc.vc_ld.models.linked_d = None*, ***kwargs*)

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

Verifiable Credential model.

class Meta

Bases: object

VerifiableCredential metadata.

schema_class = 'CredentialSchema'

add_context(*context: Union[str, dict]*)

Add a context to this credential.

add_type(*type: str*)

Add a type to this credential.

property context

Getter for context.

property context_urls: List[str]

Getter for context urls.

property credential_subject

Getter for credential subject.

property credential_subject_ids: List[str]

Getter for credential subject ids.

property expiration_date

Getter for expiration date.

property id

Getter for id.

property issuance_date

Getter for issuance date.

property issuer

Getter for issuer.

property issuer_id: Optional[str]
Getter for issuer id.

property proof
Getter for proof.

property type: List[str]
Getter for type.

class aries_cloudagent.vc.vc_ld.models.credential.**VerifiableCredentialSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Linked data verifiable credential schema.

Based on https://www.w3.org/TR/vc-data-model

proof

aries_cloudagent.vc.vc_ld.models.linked_data_proof module

LinkedDataProof.

class aries_cloudagent.vc.vc_ld.models.linked_data_proof.**LDProof**(*type: Optional[str] = None*, *proof_purpose: Optional[str] = None*, *verification_method: Optional[str] = None*, *created: Optional[str] = None*, *domain: Optional[str] = None*, *challenge: Optional[str] = None*, *jws: Optional[str] = None*, *proof_value: Optional[str] = None*, *nonce: Optional[str] = None*, ***kwargs*)

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

Linked Data Proof model.

class Meta
Bases: object

LinkedDataProof metadata.

schema_class = 'LinkedDataProofSchema'

class aries_cloudagent.vc.vc_ld.models.linked_data_proof.**LinkedDataProofSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Linked data proof schema.

Based on https://w3c-ccg.github.io/ld-proofs

class Meta
Bases: object

Accept parameter overload.

model_class
alias of *aries_cloudagent.vc.vc_ld.models.linked_data_proof.LDProof*

add_unknown_properties(*data: dict*, *original*, ***kwargs*)
Add back unknown properties before outputting.

challenge

created

domain

jws

nonce

proof_purpose

proof_value

type

verification_method

Submodules

aries_cloudagent.vc.vc_ld.issue module

Verifiable Credential issuance methods.

async aries_cloudagent.vc.vc_ld.issue.**issue**(***, *credential: dict*, *suite:* aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.LinkedDataProof, *document_loader: Callable[[str, dict], dict]*, *purpose: Optional[*aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.ProofPurpose*]* = *None*) → dict

Issue a verifiable credential.

Takes the base credentail document, verifies it, and adds a digital signature to it.

Parameters

- **credential** (*dict*) – Base credential document.
- **suite** (*LinkedDataProof*) – Signature suite to sign the credential with.
- **document_loader** (*DocumentLoader*) – Document loader to use
- **purpose** (*ProofPurpose, optional*) – A proof purpose instance that will match proofs to be verified and ensure they were created according to the appropriate purpose. Default to CredentialIssuancePurpose

Raises **LinkedDataProofException** – When the credential has an invalid structure OR signing fails

Returns The signed verifiable credential

Return type dict

aries_cloudagent.vc.vc_ld.prove module

Verifiable Credential and Presentation proving methods.

async aries_cloudagent.vc.vc_ld.prove.**create_presentation**(*, *credentials: List[dict]*, *presentation_id: Optional[str] = None*) → dict

Create presentation and add the credentials to it.

Will validates the structure off all credentials, but does not sign the presentation yet. Call sing_presentation to do this.

Parameters

- **credentials** (*List[dict]*) – Credentails to add to the presentation
- **presentation_id** (*str, optional*) – Id of the presentation. Defaults to None.

Raises **LinkedDataProofException** – When not all credentials have a valid structure

Returns The unsigned presentation object

Return type dict

async aries_cloudagent.vc.vc_ld.prove.**derive_credential**(*, *credential: dict*, *reveal_document: dict*, *suite: aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.Linke* *document_loader: Callable[[str, dict], dict]*) → dict

Derive new credential from the existing credential and the reveal document.

All proofs matching the signature suite type will be replaced with a derived proof. Other proofs will be discarded.

Parameters

- **credential** (*dict*) – The credential to derive the new credential from.
- **reveal_document** (*dict*) – JSON-LD frame to select which attributes to include.
- **suite** (*LinkedDataProof*) – The signature suite to use for derivation
- **document_loader** (*DocumentLoader*) – The document loader to use.

Returns The derived credential.

Return type dict

async aries_cloudagent.vc.vc_ld.prove.**sign_presentation**(*, *presentation: dict*, *suite: aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.Linke* *document_loader: Callable[[str, dict], dict]*, *purpose: Optional[aries_cloudagent.vc.ld_proofs.purposes.proof_purpose.* *= None*, *challenge: Optional[str] = None*, *domain: Optional[str] = None*) → dict

Sign the presentation with the passed signature suite.

Will set a default AuthenticationProofPurpose if no proof purpose is passed.

Parameters

- **presentation** (*dict*) – The presentation to sign
- **suite** (*LinkedDataProof*) – The signature suite to sign the presentation with
- **document_loader** (*DocumentLoader*) – Document loader to use.

- **purpose** (*ProofPurpose, optional*) – Purpose to use. Required if challenge is None
- **challenge** (*str, optional*) – Challenge to use. Required if domain is None.
- **domain** (*str, optional*) – Domain to use. Only used if purpose is None.

Raises **LinkedDataProofException** – When both purpose and challenge are not provided And when signing of the presentation fails

Returns A verifiable presentation object

Return type dict

aries_cloudagent.vc.vc_ld.validation_result module

Presentation verification and validation result classes.

class aries_cloudagent.vc.vc_ld.validation_result.**PresentationVerificationResult**(*, *verified: bool*, *presentation_result: Optional[aries_cloudagent.vc.ld = None*, *credential_results: Optional[List[aries_cloudagent. = None*, *errors: Optional[List[Exception]] = None*)

Bases: object

Presentation verification result class.

aries_cloudagent.vc.vc_ld.verify module

Verifiable Credential and Presentation verification methods.

async aries_cloudagent.vc.vc_ld.verify.**verify_credential**(*, *credential: dict*, *suites: List[aries_cloudagent.vc.ld_proofs.suites.linked_data_proof.L document_loader: Callable[[str, dict], dict]*, *purpose: Optional[aries_cloudagent.vc.ld_proofs.purposes.proof_purpos = None*) → *aries_cloudagent.vc.ld_proofs.validation_result.DocumentVe*

Verify credential structure, proof purpose and signature.

Parameters

- **credential** (*dict*) – The credential to verify
- **suites** (*List[LinkedDataProof]*) – The signature suites to verify with
- **document_loader** (*DocumentLoader*) – Document loader used for resolving of documents

- **purpose** (*ProofPurpose, optional*) – Proof purpose to use. Defaults to CredentialIssuancePurpose

Returns

The result of the verification. Verified property indicates whether the verification was successful

Return type DocumentVerificationResult

async aries_cloudagent.vc.vc_ld.verify.**verify_presentation**(*, *presentation: dict*, *suites: List[aries_cloudagent.vc.ld_proofs.suites.linked_data_pro document_loader: Callable[[str, dict], dict]*, *purpose: Optional[aries_cloudagent.vc.ld_proofs.purposes.proof_purp = None*, *challenge: Optional[str] = None*, *domain: Optional[str] = None*) → *aries_cloudagent.vc.vc_ld.validation_result.PresentationV*

Verify presentation structure, credentials, proof purpose and signature.

Parameters

- **presentation** (*dict*) – The presentation to verify
- **suites** (*List[LinkedDataProof]*) – The signature suites to verify with
- **document_loader** (*DocumentLoader*) – Document loader used for resolving of documents
- **purpose** (*ProofPurpose, optional*) – Proof purpose to use. Defaults to AuthenticationProofPurpose
- **challenge** (*str, optional*) – The challenge to use for authentication. Required if purpose is not passed, not used if purpose is passed
- **domain** (*str, optional*) – Domain to use for the authentication proof purpose. Not used if purpose is passed

Returns

The result of the verification. Verified property indicates whether the verification was successful

Return type PresentationVerificationResult

aries_cloudagent.wallet package

Abstract and Indy wallet handling.

Subpackages

aries_cloudagent.wallet.models package

Submodules

aries_cloudagent.wallet.models.wallet_record module

Wallet record.

class aries_cloudagent.wallet.models.wallet_record.**WalletRecord**(*, *wallet_id: Optional[str] = None*, *key_management_mode: Optional[str] = None*, *settings: Optional[dict] = None*, *wallet_name: Optional[str] = None*, *jwt_iat: Optional[int] = None*, ***kwargs*)

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

Represents a wallet record.

MODE_MANAGED = 'managed'

MODE_UNMANAGED = 'unmanaged'

class Meta

Bases: object

WalletRecord metadata.

schema_class = 'WalletRecordSchema'

RECORD_ID_NAME = 'wallet_id'

RECORD_TYPE = 'wallet_record'

TAG_NAMES = {'wallet_name'}

property is_managed: bool

Accessor to check if the key management mode is managed.

property record_value: dict

Accessor for the JSON record value generated for this record.

property requires_external_key: bool

Accessor to check if the wallet requires an external key.

property settings: dict

Accessor for the context settings associated with this wallet.

update_settings(*settings: dict*)

Update settings.

property wallet_dispatch_type: str

Accessor for webhook dispatch type of the wallet.

property wallet_id: str

Accessor for the ID associated with this record.

property wallet_key: Optional[str]

Accessor for the key of the wallet.

property wallet_key_derivation_method

Accessor for the key derivation method of the wallet.

property wallet_name: Optional[str]

Accessor for the name of the wallet.

property wallet_type: str

Accessor for the type of the wallet.

property wallet_webhook_urls: Sequence[str]

Accessor for webhook_urls of the wallet.

class aries_cloudagent.wallet.models.wallet_record.**WalletRecordSchema**(**args: Any*, ***kwargs: Any*)

Bases: marshmallow.

Schema to allow serialization/deserialization of record.

class Meta

Bases: object

WalletRecordSchema metadata.

model_class

alias of *aries_cloudagent.wallet.models.wallet_record.WalletRecord*

key_management_mode

settings

wallet_id

Submodules

aries_cloudagent.wallet.askar module

aries_cloudagent.wallet.base module

Wallet base class.

class aries_cloudagent.wallet.base.**BaseWallet**

Bases: abc.ABC

Abstract wallet interface.

abstract async create_local_did(*method:* aries_cloudagent.wallet.did_method.DIDMethod, *key_type:* aries_cloudagent.wallet.key_type.KeyType, *seed: Optional[str] = None*, *did: Optional[str] = None*, *metadata: Optional[dict] = None*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Create and store a new local DID.

Parameters

- **method** – The method to use for the DID
- **key_type** – The key type to use for the DID
- **seed** – Optional seed to use for DID
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns The created *DIDInfo*

async create_public_did(*method:* aries_cloudagent.wallet.did_method.DIDMethod, *key_type:* aries_cloudagent.wallet.key_type.KeyType, *seed: Optional[str] = None*, *did: Optional[str] = None*, *metadata: dict = {}*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Create and store a new public DID.

Parameters

- **seed** – Optional seed to use for DID

- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns The created *DIDInfo*

abstract async create_signing_key(*key_type:* aries_cloudagent.wallet.key_type.KeyType, *seed: Optional[str] = None*, *metadata: Optional[dict] = None*) → *aries_cloudagent.wallet.did_info.KeyInfo*

Create a new public/private signing keypair.

Parameters

- **key_type** – Key type to create
- **seed** – Optional seed allowing deterministic key creation
- **metadata** – Optional metadata to store with the keypair

Returns A *KeyInfo* representing the new record

abstract async get_local_did(*did: str*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Find info for a local DID.

Parameters **did** – The DID for which to get info

Returns A *DIDInfo* instance for the DID

abstract async get_local_did_for_verkey(*verkey: str*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Resolve a local DID from a verkey.

Parameters **verkey** – Verkey for which to get DID info

Returns A *DIDInfo* instance for the DID

abstract async get_local_dids() → Sequence[*aries_cloudagent.wallet.did_info.DIDInfo*]

Get list of defined local DIDs.

Returns A list of *DIDInfo* instances

async get_posted_dids() → Sequence[*aries_cloudagent.wallet.did_info.DIDInfo*]

Get list of defined posted DIDs.

Returns A list of *DIDInfo* instances

abstract async get_public_did() → *aries_cloudagent.wallet.did_info.DIDInfo*

Retrieve the public DID.

Returns The currently public *DIDInfo*, if any

abstract async get_signing_key(*verkey: str*) → *aries_cloudagent.wallet.did_info.KeyInfo*

Fetch info for a signing keypair.

Parameters **verkey** – The verification key of the keypair

Returns A *KeyInfo* representing the keypair

abstract async pack_message(*message: str*, *to_verkeys: Sequence[str]*, *from_verkey: Optional[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

abstract async replace_local_did_metadata(*did: str*, *metadata: dict*)
Replace the metadata associated with a local DID.

Prefer *set_did_endpoint()* to set endpoint in metadata.

Parameters

- **did** – DID for which to replace metadata
- **metadata** – The new metadata

abstract async 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

abstract async rotate_did_keypair_apply(*did: str*) → None
Apply temporary keypair as main for DID that wallet owns.

Parameters **did** – signing DID

Raises

- **WalletNotFoundError** – if wallet does not own DID
- **WalletError** – if wallet has not started key rotation

abstract async rotate_did_keypair_start(*did: str*, *next_seed: Optional[str] = None*) → str
Begin key rotation for DID that wallet owns: generate new keypair.

Parameters

- **did** – signing DID
- **next_seed** – seed for incoming ed25519 key pair (default random)

Returns The new verification key

Raises **WalletNotFoundError** – if wallet does not own DID

async set_did_endpoint(*did: str*, *endpoint: str*, *_ledger:* aries_cloudagent.ledger.base.BaseLedger, *endpoint_type: Optional[*aries_cloudagent.ledger.endpoint_type.EndpointType*] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*, *routing_keys: Optional[List[str]] = None*)
Update the endpoint for a DID in the wallet, send to ledger if public or posted.

Parameters

- **did** – DID for which to set endpoint
- **endpoint** – the endpoint to set, None to clear
- **ledger** – the ledger to which to send endpoint update if DID is public or posted
- **endpoint_type** – the type of the endpoint/service. Only endpoint_type 'endpoint' affects local wallet

abstract async set_public_did(*did: Union[str,* aries_cloudagent.wallet.did_info.DIDInfo*]*) → *aries_cloudagent.wallet.did_info.DIDInfo*
Assign the public DID.

Returns The updated *DIDInfo*

abstract async sign_message(*message: Union[List[bytes], bytes]*, *from_verkey: str*) → bytes
Sign message(s) using the private key associated with a given verkey.

Parameters

- **message** – The message(s) to sign
- **from_verkey** – Sign using the private key related to this verkey

Returns The signature

abstract async unpack_message(*enc_message: bytes*) → Tuple[str, str, str]
Unpack a message.

Parameters **enc_message** – The encrypted message

Returns (message, from_verkey, to_verkey)

Return type A tuple

abstract async verify_message(*message: Union[List[bytes], bytes]*, *signature: bytes*, *from_verkey: str*, *key_type:* aries_cloudagent.wallet.key_type.KeyType) → 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
- **key_type** – The key type to derive the signature verification algorithm from

Returns True if verified, else False

aries_cloudagent.wallet.bbs module

BBS+ crypto.

exception aries_cloudagent.wallet.bbs.**BbsException**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

Base BBS exception.

aries_cloudagent.wallet.bbs.**create_bls12381g2_keypair**(*seed: Optional[bytes] = None*) → Tuple[bytes, bytes]
Create a public and private bls12381g2 keypair from a seed value.

Parameters **seed** – Seed for keypair

Returns A tuple of (public key, secret key)

aries_cloudagent.wallet.bbs.**sign_messages_bls12381g2**(*messages: List[bytes]*, *secret: bytes*)
Sign messages using a bls12381g2 private signing key.

Parameters

- **messages** (*List[bytes]*) – The messages to sign
- **secret** (*bytes*) – The private signing key

Returns The signature

Return type bytes

aries_cloudagent.wallet.bbs.**verify_signed_messages_bls12381g2**(*messages: List[bytes]*, *signature: bytes*, *public_key: bytes*) → bool

Verify an ed25519 signed message according to a public verification key.

Parameters

- **signed** – The signed messages
- **public_key** – The public key to use in verification

Returns True if verified, else False

aries_cloudagent.wallet.crypto module

Cryptography functions used by BasicWallet.

aries_cloudagent.wallet.crypto.**add_pack_recipients**(*wrapper: aries_cloudagent.utils.jwe.JweEnvelope*, *cek: bytes*, *to_verkeys: Sequence[bytes]*, *from_secret: Optional[bytes] = None*)

Assemble the recipients block of a packed message.

Parameters

- **wrapper** – The envelope to add recipients to
- **cek** – The content encryption key
- **to_verkeys** – Verkeys of recipients
- **from_secret** – Secret to use for signing keys

Returns A tuple of (json result, key)

aries_cloudagent.wallet.crypto.**create_ed25519_keypair**(*seed: Optional[bytes] = None*) → Tuple[bytes, bytes]

Create a public and private ed25519 keypair from a seed value.

Parameters **seed** – Seed for keypair

Returns A tuple of (public key, secret key)

aries_cloudagent.wallet.crypto.**create_keypair**(*key_type: aries_cloudagent.wallet.key_type.KeyType*, *seed: Optional[bytes] = None*) → Tuple[bytes, bytes]

Create a public and private keypair from a seed value.

Parameters

- **key_type** – The type of key to generate
- **seed** – Seed for keypair

Raises **WalletError** – If the key type is not supported

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: aries_cloudagent.utils.jwe.JweEnvelope*, *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.**did_is_self_certified**(*did: str*, *verkey: str*) → bool

Check if the DID is self certified.

Parameters

- **did** – DID string
- **verkey** – VERKEY string

aries_cloudagent.wallet.crypto.**ed25519_pk_to_curve25519**(*public_key: bytes*) → bytes

Covert a public Ed25519 key to a public Curve25519 key as bytes.

aries_cloudagent.wallet.crypto.**encode_pack_message**(*message: str*, *to_verkeys: Sequence[bytes]*, *from_secret: Optional[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[*aries_cloudagent.utils.jwe.JweRecipient*]*) → 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.**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: Union[List[bytes], bytes]*, *secret: bytes*, *key_type:* aries_cloudagent.wallet.key_type.KeyType) → bytes

Sign message(s) using a private signing key.

Parameters

- **message** – The message(s) to sign
- **secret** – The private signing key
- **key_type** – The key type to derive the signature algorithm from

Returns The signature

Return type bytes

aries_cloudagent.wallet.crypto.**sign_message_ed25519**(*message: bytes*, *secret: bytes*) → bytes

Sign message using a ed25519 private signing key.

Parameters

- **messages** (*bytes*) – The message to sign
- **secret** (*bytes*) – The private signing key

Returns The signature

Return type bytes

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: Union[str, 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**(*message: Union[List[bytes], bytes]*, *signature: bytes*, *verkey: bytes*, *key_type: aries_cloudagent.wallet.key_type.KeyType*) → bool
Verify a signed message according to a public verification key.

Parameters

- **message** – The message(s) to verify
- **signature** – The signature to verify
- **verkey** – The verkey to use in verification
- **key_type** – The key type to derive the signature verification algorithm from

Returns True if verified, else False

aries_cloudagent.wallet.crypto.**verify_signed_message_ed25519**(*message: bytes*, *signature: bytes*, *verkey: bytes*) → bool
Verify an ed25519 signed message according to a public verification key.

Parameters

- **message** – The message to verify
- **signature** – The signature to verify
- **verkey** – The verkey to use in verification

Returns True if verified, else False

aries_cloudagent.wallet.did_info module

KeyInfo, DIDInfo.

class aries_cloudagent.wallet.did_info.**DIDInfo**(*did*, *verkey*, *metadata*, *method*, *key_type*)
Bases: tuple

property did
Alias for field number 0

property key_type
Alias for field number 4

property metadata
Alias for field number 2

property method
Alias for field number 3

property verkey
Alias for field number 1

class aries_cloudagent.wallet.did_info.**KeyInfo**(*verkey*, *metadata*, *key_type*)
Bases: tuple

property key_type
Alias for field number 2

property metadata
Alias for field number 1

property verkey
Alias for field number 0

aries_cloudagent.wallet.did_method module

did method.py contains registry for did methods.

class aries_cloudagent.wallet.did_method.**DIDMethod**(*name: str*, *key_types: List[*aries_cloudagent.wallet.key_type.KeyType*]*, *rotation: bool = False*, *holder_defined_did:* aries_cloudagent.wallet.did_method.HolderDefinedDid *= HolderDefinedDid.NO*)
Bases: object

Class to represent a did method.

holder_defined_did() → *aries_cloudagent.wallet.did_method.HolderDefinedDid*
Return the did derivation policy.

eg: did:key DIDs are derived from the verkey -> HolderDefinedDid.NO eg: did:web DIDs cannot be derived from key material -> HolderDefinedDid.REQUIRED

property method_name
Get method name.

property supported_key_types
Get supported key types.

supports_key_type(*key_type:* aries_cloudagent.wallet.key_type.KeyType) → bool
Check whether the current method supports the key type.

property supports_rotation
Check rotation support.

class aries_cloudagent.wallet.did_method.**DIDMethods**
Bases: object

DID Method class specifying DID methods with supported key types.

from_did(*did: str*) → *aries_cloudagent.wallet.did_method.DIDMethod*
Get DID method instance from the did url.

from_metadata(*metadata: Mapping*) → Optional[*aries_cloudagent.wallet.did_method.DIDMethod*]
Get DID method instance from metadata object.

Returns SOV if no metadata was found for backwards compatibility.

from_method(*method_name: str*) → Optional[*aries_cloudagent.wallet.did_method.DIDMethod*]
Retrieve a did method from method name.

register(*method:* aries_cloudagent.wallet.did_method.DIDMethod)
Register a new did method.

registered(*method: str*) → bool
Check for a supported method.

class aries_cloudagent.wallet.did_method.**HolderDefinedDid**(*value*)
Bases: enum.Enum

Define if a holder can specify its own did for a given method.

ALLOWED = 'allowed'

NO = 'no'

REQUIRED = 'required'

aries_cloudagent.wallet.did_parameters_validation module

Tooling to validate DID creation parameters.

class aries_cloudagent.wallet.did_parameters_validation.**DIDParametersValidation**(*did_methods:* aries_cloudagent.wallet.did_m
Bases: object

A utility class to check compatibility of provided DID creation parameters.

static validate_key_type(*method:* aries_cloudagent.wallet.did_method.DIDMethod, *key_type:* aries_cloudagent.wallet.key_type.KeyType)
Validate compatibility of the DID method with the desired key type.

validate_or_derive_did(*method:* aries_cloudagent.wallet.did_method.DIDMethod, *key_type:* aries_cloudagent.wallet.key_type.KeyType, *verkey: bytes*, *did: Optional[str]*) → str
Validate compatibility of the provided did (if any) with the given DID method.

If no DID was provided, automatically derive one for methods that support it.

aries_cloudagent.wallet.did_posture module

Ledger utilities.

class aries_cloudagent.wallet.did_posture.**DIDPosture**(*value*)
Bases: enum.Enum

Enum for DID postures: public, posted but not public, or in wallet only.

POSTED = DIDPostureSpec(moniker='posted', ordinal=1, public=False, posted=True)

PUBLIC = DIDPostureSpec(moniker='public', ordinal=0, public=True, posted=True)

WALLET_ONLY = DIDPostureSpec(moniker='wallet_only', ordinal=2, public=False, posted=False)

static get(*posture: Union[str, Mapping]*) → *aries_cloudagent.wallet.did_posture.DIDPosture*
Return enum instance corresponding to input string or DID metadata.

property metadata: Mapping
DID metadata for DID posture.

property moniker: str
Name for DID posture.

property ordinal: Mapping
public first, then posted and wallet-only.

Type Ordinal for presentation

class aries_cloudagent.wallet.did_posture.**DIDPostureSpec**(*moniker*, *ordinal*, *public*, *posted*)
Bases: tuple

property moniker
Alias for field number 0

property ordinal
Alias for field number 1

property posted
Alias for field number 3

property public
Alias for field number 2

aries_cloudagent.wallet.error module

Wallet-related exceptions.

exception aries_cloudagent.wallet.error.**WalletDuplicateError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.wallet.error.WalletError*

Duplicate record exception.

exception aries_cloudagent.wallet.error.**WalletError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.core.error.BaseError*

General wallet exception.

exception aries_cloudagent.wallet.error.**WalletNotFoundError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.wallet.error.WalletError*

Record not found exception.

exception aries_cloudagent.wallet.error.**WalletSettingsError**(**args*, *error_code: Optional[str] = None*, ***kwargs*)
Bases: *aries_cloudagent.wallet.error.WalletError*

Invalid settings exception.

aries_cloudagent.wallet.in_memory module

In-memory implementation of BaseWallet interface.

class aries_cloudagent.wallet.in_memory.**InMemoryWallet**(*profile:* aries_cloudagent.core.in_memory.profile.InMemoryProfile)

Bases: *aries_cloudagent.wallet.base.BaseWallet*

In-memory wallet implementation.

async create_local_did(*method:* aries_cloudagent.wallet.did_method.DIDMethod, *key_type:* aries_cloudagent.wallet.key_type.KeyType, *seed: Optional[str] = None*, *did: Optional[str] = None*, *metadata: Optional[dict] = None*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Create and store a new local DID.

Parameters

- **method** – The method to use for the DID
- **key_type** – The key type to use for the DID
- **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

async create_signing_key(*key_type:* aries_cloudagent.wallet.key_type.KeyType, *seed: Optional[str] = None*, *metadata: Optional[dict] = None*) → *aries_cloudagent.wallet.did_info.KeyInfo*

Create a new public/private signing keypair.

Parameters

- **seed** – Seed to use for signing key
- **metadata** – Optional metadata to store with the keypair
- **key_type** – Key type to generate. Default to ed25519

Returns A *KeyInfo* representing the new record

Raises **WalletDuplicateError** – If the resulting verkey already exists in the wallet

async get_local_did(*did: str*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Find info for a local DID.

Parameters **did** – The DID for which to get info

Returns A *DIDInfo* instance representing the found DID

Raises **WalletNotFoundError** – If the DID is not found

async get_local_did_for_verkey(*verkey: str*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Resolve a local DID from a verkey.

Parameters **verkey** – The verkey for which to get the local DID

Returns A *DIDInfo* instance representing the found DID

Raises **WalletNotFoundError** – If the verkey is not found

async get_local_dids() → Sequence[*aries_cloudagent.wallet.did_info.DIDInfo*]
Get list of defined local DIDs.

Returns A list of locally stored DIDs as *DIDInfo* instances

async get_public_did() → *aries_cloudagent.wallet.did_info.DIDInfo*
Retrieve the public DID.

Returns The currently public *DIDInfo*, if any

async get_signing_key(*verkey: str*) → *aries_cloudagent.wallet.did_info.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

async pack_message(*message: str*, *to_verkeys: Sequence[str]*, *from_verkey: Optional[str] = None*) → bytes
Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – List of verkeys for which to pack
- **from_verkey** – Sender verkey from which to pack

Returns The resulting packed message bytes

Raises **WalletError** – If the message is not provided

async replace_local_did_metadata(*did: str*, *metadata: dict*)
Replace metadata for a local DID.

Parameters

- **did** – The DID for which to replace metadata
- **metadata** – The new metadata

Raises **WalletNotFoundError** – If the DID doesn't exist

async 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

async rotate_did_keypair_apply(*did: str*) → None
Apply temporary keypair as main for DID that wallet owns.

Parameters **did** – signing DID

Raises

- **WalletNotFoundError** – if wallet does not own DID
- **WalletError** – if wallet has not started key rotation

async rotate_did_keypair_start(*did: str*, *next_seed: Optional[str] = None*) → str
Begin key rotation for DID that wallet owns: generate new keypair.

Parameters

- **did** – signing DID
- **next_seed** – incoming replacement seed (default random)

Returns The new verification key

Raises **WalletNotFoundError** – if wallet does not own DID

async set_public_did(*did: Union[str,* aries_cloudagent.wallet.did_info.DIDInfo*]*) → *aries_cloudagent.wallet.did_info.DIDInfo*
Assign the public DID.

Returns The updated *DIDInfo*

async sign_message(*message: Union[List[bytes], bytes]*, *from_verkey: str*) → bytes
Sign message(s) using the private key associated with a given verkey.

Parameters

- **message** – Message(s) 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

async unpack_message(*enc_message: bytes*) → Tuple[str, str, 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

async verify_message(*message: Union[List[bytes], bytes]*, *signature: bytes*, *from_verkey: str*, *key_type:* aries_cloudagent.wallet.key_type.KeyType) → bool
Verify a signature against the public key of the signer.

Parameters

- **message** – Message(s) to verify
- **signature** – Signature to verify
- **from_verkey** – Verkey to use in verification
- **key_type** – The key type to derive the signature verification algorithm from

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.indy module

Indy implementation of BaseWallet interface.

class aries_cloudagent.wallet.indy.**IndySdkWallet**(*opened:* aries_cloudagent.indy.sdk.wallet_setup.IndyOpenWallet)

Bases: *aries_cloudagent.wallet.base.BaseWallet*

Indy identity wallet implementation.

async create_local_did(*method:* aries_cloudagent.wallet.did_method.DIDMethod, *key_type:* aries_cloudagent.wallet.key_type.KeyType, *seed: Optional[str] = None*, *did: Optional[str] = None*, *metadata: Optional[dict] = None*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Create and store a new local DID.

Parameters

- **method** – The method to use for the DID
- **key_type** – The key type to use for the DID
- **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

async create_signing_key(*key_type:* aries_cloudagent.wallet.key_type.KeyType, *seed: Optional[str] = None*, *metadata: Optional[dict] = None*) → *aries_cloudagent.wallet.did_info.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

async classmethod generate_wallet_key(*seed: Optional[str] = None*) → str

Generate a raw Indy wallet key.

async get_local_did(*did: str*) → *aries_cloudagent.wallet.did_info.DIDInfo*
Find info for a local DID.

Parameters **did** – The DID for which to get info

Returns A *DIDInfo* instance representing the found DID

Raises

- **WalletNotFoundError** – If the DID is not found
- **WalletError** – If there is a libindy error

async get_local_did_for_verkey(*verkey: str*) → *aries_cloudagent.wallet.did_info.DIDInfo*
Resolve a local DID from a verkey.

Parameters **verkey** – The verkey for which to get the local DID

Returns A *DIDInfo* instance representing the found DID

Raises **WalletNotFoundError** – If the verkey is not found

async get_local_dids() → Sequence[*aries_cloudagent.wallet.did_info.DIDInfo*]
Get list of defined local DIDs.

Returns A list of locally stored DIDs as *DIDInfo* instances

async get_public_did() → *aries_cloudagent.wallet.did_info.DIDInfo*
Retrieve the public DID.

Returns The currently public *DIDInfo*, if any

async get_signing_key(*verkey: str*) → *aries_cloudagent.wallet.did_info.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

async pack_message(*message: str*, *to_verkeys: Sequence[str]*, *from_verkey: Optional[str] = None*) → bytes
Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – List of verkeys for which to pack
- **from_verkey** – Sender verkey from which to pack

Returns The resulting packed message bytes

Raises

- **WalletError** – If no message is provided
- **WalletError** – If a libindy error occurs

async replace_local_did_metadata(*did: str*, *metadata: dict*)
Replace metadata for a local DID.

Parameters

- **did** – The DID for which to replace metadata
- **metadata** – The new metadata

async 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

async rotate_did_keypair_apply(*did: str*) → *aries_cloudagent.wallet.did_info.DIDInfo*
Apply temporary keypair as main for DID that wallet owns.

Parameters **did** – signing DID

Returns DIDInfo with new verification key and metadata for DID

async rotate_did_keypair_start(*did: str*, *next_seed: Optional[str] = None*) → str
Begin key rotation for DID that wallet owns: generate new keypair.

Parameters

- **did** – signing DID
- **next_seed** – incoming replacement seed (default random)

Returns The new verification key

async set_did_endpoint(*did: str*, *endpoint: str*, *ledger:* aries_cloudagent.ledger.base.BaseLedger, *endpoint_type: Optional[*aries_cloudagent.ledger.endpoint_type.EndpointType*] = None*, *write_ledger: bool = True*, *endorser_did: Optional[str] = None*, *routing_keys: Optional[List[str]] = None*)
Update the endpoint for a DID in the wallet, send to ledger if public or posted.

Parameters

- **did** – DID for which to set endpoint
- **endpoint** – the endpoint to set, None to clear
- **ledger** – the ledger to which to send endpoint update if DID is public or posted
- **endpoint_type** – the type of the endpoint/service. Only endpoint_type 'endpoint' affects local wallet

async set_public_did(*did: Union[str,* aries_cloudagent.wallet.did_info.DIDInfo*]*) → *aries_cloudagent.wallet.did_info.DIDInfo*
Assign the public DID.

Returns The updated *DIDInfo*

async 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

async unpack_message(*enc_message: bytes*) → Tuple[str, str, 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

async verify_message(*message: Union[List[bytes], bytes]*, *signature: bytes*, *from_verkey: str*, *key_type: aries_cloudagent.wallet.key_type.KeyType*) → 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.key_pair module

Key pair storage manager.

class aries_cloudagent.wallet.key_pair.**KeyPairStorageManager**(*store: aries_cloudagent.storage.base.BaseStorage*)

Bases: object

Key pair storage manager.

async delete_key_pair(*verkey: str*)
Remove a previously-stored key pair record.

Raises **StorageNotFoundError** – If the record is not found

async find_key_pairs(*tag_query: Optional[Mapping] = None*) → List[dict]
Find key pairs by tag query.

async get_key_pair(*verkey: str*) → dict
Retrieve signing key pair from storage by verkey.

Parameters

- **storage** (*BaseStorage*) – The storage to use for querying
- **verkey** (*str*) – The verkey to query for

Raises

- **StorageDuplicateError** – If more than one key pair is found for this verkey
- **StorageNotFoundError** – If no key pair is found for this verkey

Returns dict: The key pair data

async store_key_pair(*public_key: bytes*, *secret_key: bytes*, *key_type: aries_cloudagent.wallet.key_type.KeyType*, *metadata: dict = {}*, *tags: dict = {}*)

Store signing key pair in storage.

Parameters

- **public_key** (*bytes*) – The public key
- **secret_key** (*bytes*) – The secret key
- **key_type** (*KeyType*) – The key type
- **metadata** (*dict, optional*) – The metadata
- **tags** (*dict, optional*) – The tags.

async update_key_pair_metadata(*verkey: str*, *metadata: dict*)

Update the metadata of a key pair record by verkey.

Raises **StorageNotFoundError** – If the record is not found.

aries_cloudagent.wallet.key_type module

Key type code.

class aries_cloudagent.wallet.key_type.**KeyType**(*key_type: str*, *multicodec_name: str*, *multicodec_prefix: bytes*)

Bases: object

Key Type class.

property key_type: str

Get Key type, type.

property multicodec_name: str

Get key type multicodec name.

property multicodec_prefix: bytes

Get key type multicodec prefix.

class aries_cloudagent.wallet.key_type.**KeyTypes**

Bases: object

KeyType class specifying key types with multicodec name.

from_key_type(*key_type: str*) → Optional[*aries_cloudagent.wallet.key_type.KeyType*]

Get KeyType instance from the key type identifier.

from_multicodec_name(*multicodec_name: str*) → Optional[*aries_cloudagent.wallet.key_type.KeyType*]

Get KeyType instance based on multicodec name. Returns None if not found.

from_multicodec_prefix(*multicodec_prefix: bytes*) → Optional[*aries_cloudagent.wallet.key_type.KeyType*]
Get KeyType instance based on multicodec prefix. Returns None if not found.

from_prefixed_bytes(*prefixed_bytes: bytes*) → Optional[*aries_cloudagent.wallet.key_type.KeyType*]
Get KeyType instance based on prefix in bytes. Returns None if not found.

register(*key_type:* aries_cloudagent.wallet.key_type.KeyType)
Register a new key type.

aries_cloudagent.wallet.routes module

Wallet admin routes.

class aries_cloudagent.wallet.routes.**AttribConnIdMatchInfoSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Path parameters and validators for request taking connection id.

conn_id

class aries_cloudagent.wallet.routes.**CreateAttribTxnForEndorserOptionSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Class for user to input whether to create a transaction for endorser or not.

create_transaction_for_endorser

class aries_cloudagent.wallet.routes.**DIDCreateOptionsSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Parameters and validators for create DID options.

did

key_type

class aries_cloudagent.wallet.routes.**DIDCreateSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Parameters and validators for create DID endpoint.

method

options

seed

class aries_cloudagent.wallet.routes.**DIDEndpointSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Request schema to set DID endpoint; response schema to get DID endpoint.

did

endpoint

class aries_cloudagent.wallet.routes.**DIDEndpointWithTypeSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Request schema to set DID endpoint of particular type.

did

endpoint

endpoint_type

class aries_cloudagent.wallet.routes.**DIDListQueryStringSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Parameters and validators for DID list request query string.

did

key_type

method

posture

verkey

class aries_cloudagent.wallet.routes.**DIDListSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Result schema for connection list.

results

class aries_cloudagent.wallet.routes.**DIDQueryStringSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Parameters and validators for set public DID request query string.

did

class aries_cloudagent.wallet.routes.**DIDResultSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Result schema for a DID.

result

class aries_cloudagent.wallet.routes.**DIDSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Result schema for a DID.

did

key_type

method

posture

verkey

class aries_cloudagent.wallet.routes.**MediationIDSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Class for user to optionally input a mediation_id.

mediation_id

class aries_cloudagent.wallet.routes.**WalletModuleResponseSchema**(**args: Any*, ***kwargs: Any*)
Bases: marshmallow.

Response schema for Wallet Module.

aries_cloudagent.wallet.routes.**format_did_info**(*info:* aries_cloudagent.wallet.did_info.DIDInfo)
Serialize a DIDInfo object.

async aries_cloudagent.wallet.routes.**on_register_nym_event**(*profile:* aries_cloudagent.core.profile.Profile, *event:* aries_cloudagent.core.event_bus.Event)

Handle any events we need to support.

aries_cloudagent.wallet.routes.**post_process_routes**(*app: aiohttp.web.Application*)

Amend swagger API.

async aries_cloudagent.wallet.routes.**promote_wallet_public_did**(*profile:* aries_cloudagent.core.profile.Profile, *context:* aries_cloudagent.admin.request_context.AdminRequ *session_fn*, *did: str*, *write_ledger:* *bool* = *False*, *connection_id:* *Optional[str]* = *None*, *routing_keys: Optional[List[str]]* = *None*, *mediator_endpoint:* *Optional[str]* = *None*) → *aries_cloudagent.wallet.did_info.DIDInfo*

Promote supplied DID to the wallet public DID.

async aries_cloudagent.wallet.routes.**register**(*app: aiohttp.web.Application*)

Register routes.

aries_cloudagent.wallet.routes.**register_events**(*event_bus:* aries_cloudagent.core.event_bus.EventBus)

Subscribe to any events we need to support.

aries_cloudagent.wallet.util module

Wallet utility functions.

aries_cloudagent.wallet.util.**abbr_verkey**(*full_verkey: str*, *did: Optional[str]* = *None*) → str

Given a full verkey and DID, return the abbreviated verkey.

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*, *encoding: str* = *'ascii'*) → str

Convert a byte string to base 64.

aries_cloudagent.wallet.util.**default_did_from_verkey**(*verkey: str*) → str

Given a verkey, return the default indy did.

By default the did is the first 16 bytes of the verkey.

aries_cloudagent.wallet.util.**full_verkey**(*did: str*, *abbr_verkey: str*) → str

Given a DID and abbreviated verkey, return the full verkey.

async aries_cloudagent.wallet.util.**notify_endorse_did_attrib_event**(*profile:* aries_cloudagent.core.profile.Profile, *did: str*, *meta_data: dict*)

Send notification for a DID ATTRIB post-process event.

async aries_cloudagent.wallet.util.**notify_endorse_did_event**(*profile:* aries_cloudagent.core.profile.Profile, *did: str*, *meta_data: dict*)

Send notification for a DID post-process event.

aries_cloudagent.wallet.util.**pad**(*val: str*) → str

Pad base64 values if need be: JWT calls to omit trailing padding.

aries_cloudagent.wallet.util.**random_seed**() → bytes

Generate a random seed value.

Returns A new random seed

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.

1.1.2 Submodules

1.1.3 aries_cloudagent.version module

Library version information.

CHAPTER

TWO

INDICES AND TABLES

- genindex

PYTHON MODULE INDEX

a

INDEX

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z