Aries Cloud Agent Python Documentation

See Contributors list on GitHub

Jun 30, 2022

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

aries_cloudagent.askar.didcomm.v2 module

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

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

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

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

aries_cloudagent.connections.models.connection_target module

Submodules

aries_cloudagent.connections.base_manager module

aries_cloudagent.connections.util module

aries_cloudagent.core package

Subpackages

aries_cloudagent.core.in_memory package

Subpackages

aries_cloudagent.core.in_memory.didcomm package

Submodules

aries_cloudagent.core.in_memory.didcomm.derive_1pu module

aries_cloudagent.core.in_memory.didcomm.derive_ecdh module

Submodules

aries_cloudagent.core.in_memory.profile module

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

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.

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.did package

Submodules

aries_cloudagent.did.did_key module

aries_cloudagent.holder package

Submodules

aries_cloudagent.holder.routes module

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

aries_cloudagent.indy.models.cred_abstract module

aries_cloudagent.indy.models.cred_def module

aries_cloudagent.indy.models.cred_precis module

aries_cloudagent.indy.models.cred_request module

aries_cloudagent.indy.models.non_rev_interval module

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

aries_cloudagent.indy.models.proof_request module

aries_cloudagent.indy.models.requested_creds module

aries_cloudagent.indy.models.revocation module

aries_cloudagent.indy.models.schema module

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*, *cred_ex_id: str*, *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 cred_ex_id: credential exchange identifier to use in issuer cred rev rec 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.util.**tails_path**(*rev_reg_id: str*) → str

Return path to indy tails file for input rev reg id.

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

aries_cloudagent.ledger.multiple_ledger.indy_manager module

aries_cloudagent.ledger.multiple_ledger.indy_vdr_manager module

aries_cloudagent.ledger.multiple_ledger.ledger_config_schema module

aries_cloudagent.ledger.multiple_ledger.ledger_requests_executor module

aries_cloudagent.ledger.multiple_ledger.manager_provider module

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.

abstract 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

abstract 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_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 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*)
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*)
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: bool*, *sign_did: aries_cloudagent.wallet.did_info.DIDInfo = <object object>*) → 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*) → 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:* aries_cloudagent.core.profile.Profile)

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

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*)
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.

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*)
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*)
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>*) → 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*) → 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 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 creating the schema
- **schema_name** – The schema name
- **schema_version** – The schema version

- **attribute_names** – A list of schema attributes

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*)
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.

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*)

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*)

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: bool*, *sign_did: aries_cloudagent.wallet.did_info.DIDInfo = <object object>*) → 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*) → 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

aries_cloudagent.messaging.decorators package

Submodules

aries_cloudagent.messaging.decorators.attach_decorator module

aries_cloudagent.messaging.decorators.base module

aries_cloudagent.messaging.decorators.default module

aries_cloudagent.messaging.decorators.localization_decorator module

aries_cloudagent.messaging.decorators.please_ack_decorator module

aries_cloudagent.messaging.decorators.service_decorator module

aries_cloudagent.messaging.decorators.signature_decorator module

aries_cloudagent.messaging.decorators.thread_decorator module

aries_cloudagent.messaging.decorators.timing_decorator module

aries_cloudagent.messaging.decorators.trace_decorator module

aries_cloudagent.messaging.decorators.transport_decorator module

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

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

aries_cloudagent.messaging.models package

Common code for messaging models.

Submodules

aries_cloudagent.messaging.models.base module

aries_cloudagent.messaging.models.base_record module

aries_cloudagent.messaging.models.openapi module

aries_cloudagent.messaging.schemas package

Submodules

aries_cloudagent.messaging.schemas.routes module

aries_cloudagent.messaging.schemas.util module

Submodules

aries_cloudagent.messaging.agent_message module

aries_cloudagent.messaging.base_handler module

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

aries_cloudagent.messaging.responder module

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

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.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

aries_cloudagent.protocols.actionmenu.v1_0.handlers.menu_request_handler module

aries_cloudagent.protocols.actionmenu.v1_0.handlers.perform_handler module

aries_cloudagent.protocols.actionmenu.v1_0.messages package

Submodules

aries_cloudagent.protocols.actionmenu.v1_0.messages.menu module

aries_cloudagent.protocols.actionmenu.v1_0.messages.menu_request module

aries_cloudagent.protocols.actionmenu.v1_0.messages.perform module

aries_cloudagent.protocols.actionmenu.v1_0.models package

Submodules

aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form module

aries_cloudagent.protocols.actionmenu.v1_0.models.menu_form_param module

aries_cloudagent.protocols.actionmenu.v1_0.models.menu_option module

Submodules

aries_cloudagent.protocols.actionmenu.v1_0.base_service module

aries_cloudagent.protocols.actionmenu.v1_0.controller module

aries_cloudagent.protocols.actionmenu.v1_0.driver_service module

aries_cloudagent.protocols.actionmenu.v1_0.message_types module

Message type identifiers for Action Menus.

aries_cloudagent.protocols.actionmenu.v1_0.routes module

aries_cloudagent.protocols.actionmenu.v1_0.util module

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

aries_cloudagent.protocols.basicmessage.v1_0.messages package

Submodules

aries_cloudagent.protocols.basicmessage.v1_0.messages.basicmessage module

Submodules

aries_cloudagent.protocols.basicmessage.v1_0.message_types module

Message type identifiers for Connections.

aries_cloudagent.protocols.basicmessage.v1_0.routes module

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

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

aries_cloudagent.protocols.connections.v1_0.messages.connection_request module

aries_cloudagent.protocols.connections.v1_0.messages.connection_response module

aries_cloudagent.protocols.connections.v1_0.messages.problem_report module

aries_cloudagent.protocols.connections.v1_0.models package

Submodules

aries_cloudagent.protocols.connections.v1_0.models.connection_detail module

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

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_query_handler module

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_handler module

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.keylist_update_response_handler module

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_deny_handler module

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_grant_handler module

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.mediation_request_handler module

aries_cloudagent.protocols.coordinate_mediation.v1_0.handlers.problem_report_handler module

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

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_query_paginate module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_update_rule module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.inner.keylist_updated module

Submodules

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_query module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.keylist_update_response module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_deny module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_grant module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.mediate_request module

aries_cloudagent.protocols.coordinate_mediation.v1_0.messages.problem_report module

aries_cloudagent.protocols.coordinate_mediation.v1_0.models package

Submodules

aries_cloudagent.protocols.coordinate_mediation.v1_0.models.mediation_record module

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

aries_cloudagent.protocols.coordinate_mediation.v1_0.message_types module

Message type identifiers for Coordinate Mediation protocol.

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.Mea*

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

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

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

aries_cloudagent.protocols.didexchange.v1_0.messages.response module

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

aries_cloudagent.protocols.discovery.v1_0.handlers.query_handler module

aries_cloudagent.protocols.discovery.v1_0.messages package

Submodules

aries_cloudagent.protocols.discovery.v1_0.messages.disclose module

aries_cloudagent.protocols.discovery.v1_0.messages.query module

aries_cloudagent.protocols.discovery.v1_0.models package

Package-wide code and data.

Submodules

aries_cloudagent.protocols.discovery.v1_0.models.discovery_record module

Submodules

aries_cloudagent.protocols.discovery.v1_0.manager module

aries_cloudagent.protocols.discovery.v1_0.message_types module

Message type identifiers for Feature Discovery.

aries_cloudagent.protocols.discovery.v1_0.routes module

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

aries_cloudagent.protocols.discovery.v2_0.handlers.queries_handler module

aries_cloudagent.protocols.discovery.v2_0.messages package

Submodules

aries_cloudagent.protocols.discovery.v2_0.messages.disclosures module

aries_cloudagent.protocols.discovery.v2_0.messages.queries module

aries_cloudagent.protocols.discovery.v2_0.models package

Package-wide code and data.

Submodules

aries_cloudagent.protocols.discovery.v2_0.models.discovery_record module

Submodules

aries_cloudagent.protocols.discovery.v2_0.manager module

aries_cloudagent.protocols.discovery.v2_0.message_types module

Message type identifiers for Feature Discovery.

aries_cloudagent.protocols.discovery.v2_0.routes module

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

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.refused_transaction_response_handler module

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_acknowledgement_handler module

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_cancel_handler module

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_job_to_send_handler module

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_request_handler module

aries_cloudagent.protocols.endorse_transaction.v1_0.handlers.transaction_resend_handler module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages package

Submodules

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.cancel_transaction module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.endorsed_transaction_response module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.messages_attach module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.refused_transaction_response module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_acknowledgement module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_job_to_send module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_request module

aries_cloudagent.protocols.endorse_transaction.v1_0.messages.transaction_resend module

aries_cloudagent.protocols.endorse_transaction.v1_0.models package

Submodules

aries_cloudagent.protocols.endorse_transaction.v1_0.models.transaction_record module

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

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

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

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

aries_cloudagent.protocols.introduction.v0_1.messages.invitation module

aries_cloudagent.protocols.introduction.v0_1.messages.invitation_request module

Submodules

aries_cloudagent.protocols.introduction.v0_1.base_service module

aries_cloudagent.protocols.introduction.v0_1.demo_service module

aries_cloudagent.protocols.introduction.v0_1.message_types module

Message type identifiers for Introductions.

aries_cloudagent.protocols.introduction.v0_1.routes module

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

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

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_ack module

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_issue module

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_offer module

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_problem_report module

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_proposal module

aries_cloudagent.protocols.issue_credential.v1_0.messages.credential_request module

aries_cloudagent.protocols.issue_credential.v1_0.models package

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.models.credential_exchange module

Submodules

aries_cloudagent.protocols.issue_credential.v1_0.controller module

aries_cloudagent.protocols.issue_credential.v1_0.manager module

aries_cloudagent.protocols.issue_credential.v1_0.message_types module

aries_cloudagent.protocols.issue_credential.v1_0.routes module

aries_cloudagent.protocols.issue_credential.v2_0 package

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

aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.models.cred_detail_options module

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.formats.ld_proof.handler module

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.formats.handler module

aries_cloudagent.protocols.issue_credential.v2_0.handlers package

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_ack_handler module

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_issue_handler module

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_offer_handler module

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_problem_report_handler module

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_proposal_handler module

aries_cloudagent.protocols.issue_credential.v2_0.handlers.cred_request_handler module

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

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_ack module

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_format module

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_issue module

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_offer module

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_problem_report module

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_proposal module

aries_cloudagent.protocols.issue_credential.v2_0.messages.cred_request module

aries_cloudagent.protocols.issue_credential.v2_0.models package

Subpackages

aries_cloudagent.protocols.issue_credential.v2_0.models.detail package

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.models.detail.indy module

aries_cloudagent.protocols.issue_credential.v2_0.models.detail.ld_proof module

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.models.cred_ex_record module

Submodules

aries_cloudagent.protocols.issue_credential.v2_0.controller module

aries_cloudagent.protocols.issue_credential.v2_0.manager module

aries_cloudagent.protocols.issue_credential.v2_0.message_types module

aries_cloudagent.protocols.issue_credential.v2_0.routes module

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

aries_cloudagent.protocols.notification.v1_0.messages package

Submodules

aries_cloudagent.protocols.notification.v1_0.messages.ack module

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

aries_cloudagent.protocols.out_of_band.v1_0.messages.problem_report module

aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse module

aries_cloudagent.protocols.out_of_band.v1_0.messages.reuse_accept module

aries_cloudagent.protocols.out_of_band.v1_0.messages.service module

aries_cloudagent.protocols.out_of_band.v1_0.models package

Submodules

aries_cloudagent.protocols.out_of_band.v1_0.models.invitation module

aries_cloudagent.protocols.out_of_band.v1_0.models.oob_record module

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

aries_cloudagent.protocols.present_proof.dif.pres_exch_handler module

aries_cloudagent.protocols.present_proof.dif.pres_proposal_schema module

aries_cloudagent.protocols.present_proof.dif.pres_request_schema module

aries_cloudagent.protocols.present_proof.dif.pres_schema module

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

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_ack module

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_problem_report module

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_proposal module

aries_cloudagent.protocols.present_proof.v1_0.messages.presentation_request module

aries_cloudagent.protocols.present_proof.v1_0.models package

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

aries_cloudagent.protocols.present_proof.v1_0.routes module

aries_cloudagent.protocols.present_proof.v2_0 package

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

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

aries_cloudagent.protocols.present_proof.v2_0.handlers package

Submodules

aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_ack_handler module

aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_handler module

aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_problem_report_handler module

aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_proposal_handler module

aries_cloudagent.protocols.present_proof.v2_0.handlers.pres_request_handler module

aries_cloudagent.protocols.present_proof.v2_0.messages package

Submodules

aries_cloudagent.protocols.present_proof.v2_0.messages.pres module

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_ack module

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_format module

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_problem_report module

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_proposal module

aries_cloudagent.protocols.present_proof.v2_0.messages.pres_request module

aries_cloudagent.protocols.present_proof.v2_0.models package

Submodules

aries_cloudagent.protocols.present_proof.v2_0.models.pres_exchange module

Submodules

aries_cloudagent.protocols.present_proof.v2_0.controller module

aries_cloudagent.protocols.present_proof.v2_0.manager module

aries_cloudagent.protocols.present_proof.v2_0.message_types module

aries_cloudagent.protocols.present_proof.v2_0.routes module

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

Submodules

aries_cloudagent.protocols.problem_report.v1_0.handler module

aries_cloudagent.protocols.problem_report.v1_0.message module

aries_cloudagent.protocols.problem_report.v1_0.message_types module

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

aries_cloudagent.protocols.revocation_notification.v1_0.messages package

Submodules

aries_cloudagent.protocols.revocation_notification.v1_0.messages.revoke module

aries_cloudagent.protocols.revocation_notification.v1_0.models package

Submodules

aries_cloudagent.protocols.revocation_notification.v1_0.models.rev_notification_record module

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

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

aries_cloudagent.protocols.revocation_notification.v2_0.messages package

Submodules

aries_cloudagent.protocols.revocation_notification.v2_0.messages.revoke module

aries_cloudagent.protocols.revocation_notification.v2_0.models package

Submodules

aries_cloudagent.protocols.revocation_notification.v2_0.models.rev_notification_record module

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

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

aries_cloudagent.protocols.routing.v1_0.handlers.route_query_response_handler module

aries_cloudagent.protocols.routing.v1_0.handlers.route_update_request_handler module

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

aries_cloudagent.protocols.routing.v1_0.messages.route_query_request module

aries_cloudagent.protocols.routing.v1_0.messages.route_query_response module

aries_cloudagent.protocols.routing.v1_0.messages.route_update_request module

aries_cloudagent.protocols.routing.v1_0.messages.route_update_response module

aries_cloudagent.protocols.routing.v1_0.models package

Submodules

aries_cloudagent.protocols.routing.v1_0.models.paginate module

aries_cloudagent.protocols.routing.v1_0.models.paginated module

aries_cloudagent.protocols.routing.v1_0.models.route_query_result module

aries_cloudagent.protocols.routing.v1_0.models.route_record module

aries_cloudagent.protocols.routing.v1_0.models.route_update module

aries_cloudagent.protocols.routing.v1_0.models.route_updated module

Submodules

aries_cloudagent.protocols.routing.v1_0.manager module

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

aries_cloudagent.protocols.trustping.v1_0.handlers.ping_response_handler module

aries_cloudagent.protocols.trustping.v1_0.messages package

Submodules

aries_cloudagent.protocols.trustping.v1_0.messages.ping module

aries_cloudagent.protocols.trustping.v1_0.messages.ping_response module

Submodules

aries_cloudagent.protocols.trustping.v1_0.message_types module

Message type identifiers for Trust Pings.

aries_cloudagent.protocols.trustping.v1_0.routes module

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

aries_cloudagent.resolver.default.web module

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]*) → 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**(*registry:* aries_cloudagent.resolver.did_resolver_registry.DIDResolver

Bases: object

did resolver singleton.

async dereference(*profile:* aries_cloudagent.core.profile.Profile, *did_url: str*, *, *cls: Type[aries_cloudagent.resolver.did_resolver.ResourceType] = pydid.Resource*) → aries_cloudagent.resolver.did_resolver.ResourceType

Dereference a DID URL to its corresponding DID Doc object.

async resolve(*profile:* aries_cloudagent.core.profile.Profile, *did: Union[str, pydid.DID]*) → 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.did_resolver_registry module

In memmory storage for registering did resolvers.

class aries_cloudagent.resolver.did_resolver_registry.**DIDResolverRegistry**

Bases: object

Registry for did resolvers.

register(*resolver*) → None

Register a resolver.

property resolvers: Sequence[*aries_cloudagent.resolver.base.BaseDIDResolver*]

Accessor for a list of all did resolvers.

aries_cloudagent.resolver.routes module

aries_cloudagent.revocation package

Subpackages

aries_cloudagent.revocation.models package

Submodules

aries_cloudagent.revocation.models.indy module

aries_cloudagent.revocation.models.issuer_cred_rev_record module

aries_cloudagent.revocation.models.issuer_rev_reg_record module

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

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

aries_cloudagent.storage.vc_holder.in_memory module

aries_cloudagent.storage.vc_holder.indy module

aries_cloudagent.storage.vc_holder.vc_record module

aries_cloudagent.storage.vc_holder.xform module

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

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

aries_cloudagent.transport package

Subpackages

aries_cloudagent.transport.inbound package

Submodules

aries_cloudagent.transport.inbound.base module

aries_cloudagent.transport.inbound.delivery_queue module

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.MessageReceipt*, ***, *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.

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

aries_cloudagent.transport.outbound.message module

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

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

aries_cloudagent.utils.jwe module

aries_cloudagent.utils.outofband module

aries_cloudagent.utils.repeat module

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

aries_cloudagent.vc package

Subpackages

aries_cloudagent.vc.ld_proofs package

Subpackages

aries_cloudagent.vc.ld_proofs.crypto package

Submodules

aries_cloudagent.vc.ld_proofs.crypto.key_pair module

aries_cloudagent.vc.ld_proofs.crypto.wallet_key_pair module

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_timestar 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

aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_2020_base module

aries_cloudagent.vc.ld_proofs.suites.bbs_bls_signature_proof_2020 module

aries_cloudagent.vc.ld_proofs.suites.ed25519_signature_2018 module

aries_cloudagent.vc.ld_proofs.suites.jws_linked_data_signature module

aries_cloudagent.vc.ld_proofs.suites.linked_data_proof module

aries_cloudagent.vc.ld_proofs.suites.linked_data_signature module

Submodules

aries_cloudagent.vc.ld_proofs.check module

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

aries_cloudagent.vc.ld_proofs.proof_set module

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

Subpackages

aries_cloudagent.vc.vc_ld.models package

Submodules

aries_cloudagent.vc.vc_ld.models.credential module

aries_cloudagent.vc.vc_ld.models.linked_data_proof module

Submodules

aries_cloudagent.vc.vc_ld.issue module

aries_cloudagent.vc.vc_ld.prove module

aries_cloudagent.vc.vc_ld.validation_result module

aries_cloudagent.vc.vc_ld.verify module

aries_cloudagent.wallet package

Abstract and Indy wallet handling.

Subpackages

aries_cloudagent.wallet.models package

Submodules

aries_cloudagent.wallet.models.wallet_record module

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*)

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

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 enum.

class aries_cloudagent.wallet.did_method.DIDMethod(*value*)
Bases: enum.Enum

DID Method class specifying DID methods with supported key types.

KEY = DIDMethodSpec(method_name='key', supported_key_types=[<KeyType.ED25519: KeySpec(key_type='ed25519', multicodec_name='ed25519-pub', multicodec_prefix=b'\xed\x01')>, <KeyType.BLS12381G2: KeySpec(key_type='bls12381g2', multicodec_name='bls12_381-g2-pub', multicodec_prefix=b'\xeb\x01')>], supports_rotation=False)

SOV = DIDMethodSpec(method_name='sov', supported_key_types=[<KeyType.ED25519: KeySpec(key_type='ed25519', multicodec_name='ed25519-pub', multicodec_prefix=b'\xed\x01')>], supports_rotation=True)

from_did() → *aries_cloudagent.wallet.did_method.DIDMethod*
Get DID method instance from the method name.

from_metadata() → *aries_cloudagent.wallet.did_method.DIDMethod*
Get DID method instance from metadata object.

Returns SOV if no metadata was found for backwards compatability.

from_method() → Optional[*aries_cloudagent.wallet.did_method.DIDMethod*]
Get DID method instance from the method name.

property method_name: str
Getter for did method name. e.g. sov or key.

property supported_key_types: List[*aries_cloudagent.wallet.key_type.KeyType*]
Getter for supported key types of method.

supports_key_type(*key_type:* aries_cloudagent.wallet.key_type.KeyType) → bool
Check whether the current method supports the key type.

property supports_rotation: bool
Check whether the current method supports key rotation.

class aries_cloudagent.wallet.did_method.DIDMethodSpec(*method_name*, *supported_key_types*, *supports_rotation*)
Bases: tuple

property method_name
Alias for field number 0

property supported_key_types
Alias for field number 1

property supports_rotation
Alias for field number 2

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

aries_cloudagent.wallet.indy module

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 enum.

class aries_cloudagent.wallet.key_type.**KeySpec**(*key_type*, *multicodec_name*, *multicodec_prefix*)
Bases: tuple

property key_type
Alias for field number 0

property multicodec_name
Alias for field number 1

property multicodec_prefix
Alias for field number 2

class aries_cloudagent.wallet.key_type.**KeyType**(*value*)
Bases: enum.Enum

KeyType Enum specifying key types with multicodec name.

BLS12381G1 = KeySpec(key_type='bls12381g1', multicodec_name='bls12_381-g1-pub', multicodec_prefix=b'\xea\x01')

BLS12381G1G2 = KeySpec(key_type='bls12381g1g2', multicodec_name='bls12_381-g1g2-pub', multicodec_prefix=b'\xee\x01')

BLS12381G2 = KeySpec(key_type='bls12381g2', multicodec_name='bls12_381-g2-pub', multicodec_prefix=b'\xeb\x01')

ED25519 = KeySpec(key_type='ed25519', multicodec_name='ed25519-pub', multicodec_prefix=b'\xed\x01')

X25519 = KeySpec(key_type='x25519', multicodec_name='x25519-pub', multicodec_prefix=b'\xec\x01')

classmethod from_key_type(*key_type: str*) → Optional[*aries_cloudagent.wallet.key_type.KeyType*]
Get KeyType instance from the key type identifier.

classmethod 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.

classmethod 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.

classmethod 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.

property key_type: str
Getter for key type identifier.

property multicodec_name: str
Getter for multicodec name.

property multicodec_prefix: bytes
Getter for multicodec prefix.

exception aries_cloudagent.wallet.key_type.**KeyTypeException**
Bases: BaseException

Key type exception.

aries_cloudagent.wallet.routes module

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_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

J

K

L

M

N

Q

R

S

T

U

V

W

X

Y