
Aries Cloud Agent Python Documentation

Nicholas Rempel, Andrew Whitehead

Dec 13, 2019

Aries Cloud Agent Python - Modules

1	aries_cloudagent package	3
1.1	Subpackages	3
1.2	Submodules	92
1.3	aries_cloudagent.classloader module	92
1.4	aries_cloudagent.conductor module	92
1.5	aries_cloudagent.defaults module	92
1.6	aries_cloudagent.dispatcher module	92
1.7	aries_cloudagent.error module	92
1.8	aries_cloudagent.postgres module	92
1.9	aries_cloudagent.stats module	92
1.10	aries_cloudagent.task_processor module	92
1.11	aries_cloudagent.version module	92
2	Indices and tables	93
	Python Module Index	95
	Index	97

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](#).

All of the code in ACA-Py is within the [aries_cloudagent](#) module. To investigate the code, use search or click on the *aries_cloudagent package* link in the left menu to drill down 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 [messaging](#) subpackage. Other general purpose subpackages that might be of particular 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.

aries_cloudagent package

Aries Cloud Agent.

1.1 Subpackages

1.1.1 aries_cloudagent.admin package

Submodules

aries_cloudagent.admin.base_server module

Abstract admin server interface.

```
class aries_cloudagent.admin.base_server.BaseAdminServer
```

Bases: `abc.ABC`

Admin HTTP server class.

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

Add a webhook target.

```
remove_webhook_target (target_url: str)
```

Remove a webhook target.

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

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

```
start () → None
```

Start the webserver.

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

```
stop () → None
```

Stop the webserver.

aries_cloudagent.admin.error module

Admin error classes.

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

Bases: aries_cloudagent.core.error.BaseError

Base class for Admin-related errors.

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

Bases: aries_cloudagent.admin.error.AdminError

Admin server setup or configuration error.

aries_cloudagent.admin.routes module

aries_cloudagent.admin.server module

Admin server classes.

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

Bases: sphinx.ext.autodoc.importer._MockObject

Schema for the modules endpoint.

result

Used by autodoc_mock_imports.

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

Bases: aries_cloudagent.messaging.responder.BaseResponder

Handle outgoing messages from message handlers.

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

Send outbound message.

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

send_webhook(topic: str, payload: dict)

Dispatch a webhook.

Parameters

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

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

Bases: aries_cloudagent.admin.base_server.BaseAdminServer

Admin HTTP server class.

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

Add a webhook target.

make_application () → *<sphinx.ext.autodoc.importer._MockObject object at 0x7f538c670ba8>*

Get the aiohttp application instance.

on_startup (*app: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538c670ba8>*)

Perform webserver startup actions.

plugins_handler (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538c670ba8>*)

Request handler for the loaded plugins list.

Parameters **request** – aiohttp request object

Returns The module list response

redirect_handler (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538c670ba8>*)

Perform redirect to documentation.

remove_webhook_target (*target_url: str*)

Remove a webhook target.

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

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

start () → None

Start the webserver.

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

status_handler (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538c670ba8>*)

Request handler for the server status information.

Parameters **request** – aiohttp request object

Returns The web response

status_reset_handler (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538c670ba8>*)

Request handler for resetting the timing statistics.

Parameters **request** – aiohttp request object

Returns The web response

stop () → None

Stop the webserver.

websocket_handler (*request*)

Send notifications to admin client over websocket.

class `aries_cloudagent.admin.server.AdminStatusSchema` (*args, **kwargs)

Bases: `sphinx.ext.autodoc.importer._MockObject`

Schema for the status endpoint.

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

Bases: `object`

Class for managing webhook target information.

topic_filter

Accessor for the target's topic filter.

1.1.2 aries_cloudagent.cache package

Submodules

aries_cloudagent.cache.base module

Abstract base classes for cache.

class aries_cloudagent.cache.base.BaseCache

Bases: abc.ABC

Abstract cache interface.

acquire (*key: str*)

Acquire a lock on a given cache key.

clear (*key: str*)

Remove an item from the cache, if present.

Parameters **key** – the key to remove

flush ()

Remove all items from the cache.

get (*key: str*)

Get an item from the cache.

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

Returns The record found or *None*

release (*key: str*)

Release the lock on a given cache key.

set (*keys: Union[str, Sequence[str]], value: Any, ttl: int = None*)

Add an item to the cache with an optional ttl.

Parameters

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

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

Bases: aries_cloudagent.core.error.BaseError

Base class for cache-related errors.

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

Bases: object

A lock on a particular cache key.

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

done
Accessor for the done state.

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

parent
Accessor for the parent key lock, if any.

release()
Release the cache lock.

result
Fetch the current result, if any.

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

aries_cloudagent.cache.basic module

Basic in-memory cache implementation.

class `aries_cloudagent.cache.basic.BasicCache`
Bases: `aries_cloudagent.cache.base.BaseCache`
Basic in-memory cache class.

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

flush()
Remove all items from the cache.

get (*key: str*)
Get an item from the cache.
Parameters **key** – the key to retrieve an item for
Returns The record found or *None*

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

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

1.1.3 aries_cloudagent.commands package

Commands module common setup.

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

`aries_cloudagent.commands.load_command(command: str)`

Load the module corresponding with a named command.

`aries_cloudagent.commands.run_command(command: str, argv: Sequence[str] = None)`

Execute a named command with command line arguments.

Submodules

`aries_cloudagent.commands.help` module

Help command for indexing available commands.

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

Execute the help command.

`aries_cloudagent.commands.provision` module

Provision command for setting up agent settings before starting.

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

Bases: `aries_cloudagent.core.error.BaseError`

Base exception for provisioning errors.

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

Entrypoint.

`aries_cloudagent.commands.provision.init_argument_parser(parser: parse.ArgumentParser)`

Initialize an argument parser with the module's arguments.

`aries_cloudagent.commands.provision.provision(settings: dict)`

Perform provisioning.

`aries_cloudagent.commands.start` module

Entrypoint.

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

Entrypoint.

`aries_cloudagent.commands.start.init_argument_parser(parser: parse.ArgumentParser)`

Initialize an argument parser with the module's arguments.

`aries_cloudagent.commands.start.run_loop(startup: Coroutine[T_co, T_contra, V_co], shutdown: Coroutine[T_co, T_contra, V_co])`

Execute the application, handling signals and ctrl-c.

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

Shut down.

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

Start up.

1.1.4 aries_cloudagent.config package

Submodules

aries_cloudagent.config.argparse module

Command line option parsing.

```
class aries_cloudagent.config.argparse.AdminGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup
    Admin server settings.

    CATEGORIES = ('start',)
    GROUP_NAME = 'Admin'

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

    get_settings (args: argparse.Namespace)
        Extract admin settings.
```

```
class aries_cloudagent.config.argparse.ArgumentGroup
    Bases: abc.ABC
    A class representing a group of related command line arguments.

    GROUP_NAME = None

    add_arguments (parser: argparse.ArgumentParser)
        Add arguments to the provided argument parser.

    get_settings (args: argparse.Namespace) → dict
        Extract settings from the parsed arguments.
```

```
class aries_cloudagent.config.argparse.DebugGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup
    Debug settings.

    CATEGORIES = ('start',)
    GROUP_NAME = 'Debug'

    add_arguments (parser: argparse.ArgumentParser)
        Add debug command line arguments to the parser.

    get_settings (args: argparse.Namespace) → dict
        Extract debug settings.
```

```
class aries_cloudagent.config.argparse.GeneralGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup
    General settings.

    CATEGORIES = ('general', 'start')
    GROUP_NAME = 'General'

    add_arguments (parser: argparse.ArgumentParser)
        Add general command line arguments to the parser.

    get_settings (args: argparse.Namespace) → dict
        Extract general settings.
```

```
class aries_cloudagent.config.argparse.LedgerGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup

    Ledger settings.

    CATEGORIES = ('start', 'general')

    GROUP_NAME = 'Ledger'

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

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

class aries_cloudagent.config.argparse.LoggingGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup

    Logging settings.

    CATEGORIES = ('general', 'start')

    GROUP_NAME = 'Logging'

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

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

class aries_cloudagent.config.argparse.ProtocolGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup

    Protocol settings.

    CATEGORIES = ('start',)

    GROUP_NAME = 'Protocol'

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

    get_settings(args: argparse.Namespace) → dict
        Get protocol settings.

class aries_cloudagent.config.argparse.TransportGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup

    Transport settings.

    CATEGORIES = ('start',)

    GROUP_NAME = 'Transport'

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

    get_settings(args: argparse.Namespace)
        Extract transport settings.

class aries_cloudagent.config.argparse.WalletGroup
    Bases: aries_cloudagent.config.argparse.ArgumentGroup

    Wallet settings.

    CATEGORIES = ('general', 'start')
```

GROUP_NAME = 'Wallet'

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

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

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

Decorator for registering argument groups.

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

aries_cloudagent.config.argparse.load_argument_groups (*parser: argparse.ArgumentParser, *groups*)

Log a set of argument groups into a parser.

Returns A callable to convert loaded arguments into a settings dictionary

aries_cloudagent.config.base module

Configuration base classes.

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

Base injector class.

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

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

Parameters

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

Returns An instance of the base class, or None

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

Base provider class.

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

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

Base settings class.

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

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

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

Parameters

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

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

Parameters

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

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

Parameters

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

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

Parameters

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

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

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

Bases: aries_cloudagent.core.error.BaseError

A base exception for all configuration errors.

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

Bases: *aries_cloudagent.config.base.ConfigError*

The base exception raised by *BaseInjector* implementations.

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

Bases: *aries_cloudagent.config.base.ConfigError*

The base exception raised by *BaseProvider* implementations.

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

Bases: *aries_cloudagent.config.base.ConfigError*

The base exception raised by *BaseSettings* implementations.

aries_cloudagent.config.base_context module

Base injection context builder classes.


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

    Bases: abc.ABC

    Base injection context builder class.

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

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

aries_cloudagent.config.default_context module

Classes for configuring the default injection context.

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

    Bases: aries_cloudagent.config.base_context.ContextBuilder

    Default context builder.

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

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

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

aries_cloudagent.config.error module

Errors for config modules.

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

    Bases: aries_cloudagent.config.base.ConfigError

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

aries_cloudagent.config.injection_context module

Injection context implementation.

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

    Bases: aries_cloudagent.config.base.BaseInjector

    Manager for configuration settings and class providers.

    ROOT_SCOPE = 'application'
```

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

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

Parameters

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

Returns An instance of the base class, or None

injector
Accessor for scope-specific injector.

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

Parameters **scope_name** – The unique scope identifier

scope_name
Accessor for the current scope name.

settings
Accessor for scope-specific settings.

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

Parameters

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

Returns A new injection context representing the scope

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

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

Bases: *aries_cloudagent.config.base.InjectorError*

Base class for issues in the injection context.

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

injector
Alias for field number 1

name
Alias for field number 0

aries_cloudagent.config.injector module

Standard Injector implementation.

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

Bases: *aries_cloudagent.config.base.BaseInjector*

Injector implementation with static and dynamic bindings.

bind_instance (*base_cls: type, instance: object*)

Add a static instance as a class binding.

bind_provider (*base_cls: type, provider: aries_cloudagent.config.base.BaseProvider, *, cache: bool = False*)

Add a dynamic instance resolver as a class binding.

clear_binding (*base_cls: type*)

Remove a previously-added binding.

copy () → aries_cloudagent.config.base.BaseInjector

Produce a copy of the injector instance.

get_provider (*base_cls: type*)

Find the provider associated with a class binding.

inject (*base_cls: type, settings: Mapping[str, object] = None, *, required: bool = True*)

Get the provided instance of a given class identifier.

Parameters

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

Returns An instance of the base class, or None

settings

Accessor for scope-specific settings.

aries_cloudagent.config.ledger module

Ledger configuration.

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

Perform TAA acceptance.

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

Get genesis transactions.

aries_cloudagent.config.ledger.**ledger_config** (*context: aries_cloudagent.config.injection_context.InjectionContext, public_id: str, provision: bool = False*) → bool

Perform Indy ledger configuration.

aries_cloudagent.config.logging module

Utilities related to logging.

class `aries_cloudagent.config.logging.LoggingConfigurator`

Bases: `object`

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

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

Configure logger.

Parameters

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

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

Print a startup banner describing the configuration.

Parameters

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

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

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

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

Returns A file-like object representing the resource

aries_cloudagent.config.provider module

Service provider implementations.

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

Bases: `aries_cloudagent.config.base.BaseProvider`

Cache the result of another provider.

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

Provide the object instance given a config and injector.

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

Bases: `aries_cloudagent.config.base.BaseProvider`

Provider for a particular class.

```

class Inject (base_cls: type)
    Bases: object

    A class for passing injected arguments to the constructor.

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

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

    Provider for a previously-created instance.

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

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

    Add statistics to the results of another provider.

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

```

aries_cloudagent.config.settings module

Settings implementation.

```

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

    Mutable settings implementation.

    clear_value (var_name: str)
        Remove a setting.

        Parameters var_name – The name of the setting

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

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

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

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

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

        Parameters

```

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

set_value (*var_name: str, value*)

Add a setting.

Parameters

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

aries_cloudagent.config.util module

Entrypoint.

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

Bases: `object`

Argument value parser for byte sizes.

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

Perform common app configuration.

aries_cloudagent.config.wallet module

Wallet configuration.

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

Initialize the wallet.

1.1.5 aries_cloudagent.holder package

Submodules

aries_cloudagent.holder.base module

Base holder class.

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

Bases: `abc.ABC`

Base class for holder.

aries_cloudagent.holder.indy module

Indy issuer implementation.

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

Bases: `aries_cloudagent.holder.base.BaseHolder`

Indy holder class.

RECORD_TYPE_MIME_TYPES = 'attribute-mime-types'

create_credential_request (*credential_offer, credential_definition, did*)

Create a credential offer for the given credential definition id.

Parameters

- **credential_offer** – The credential offer to create request for
- **credential_definition** – The credential definition to create an offer for

Returns A credential request

create_presentation (*presentation_request: dict, requested_credentials: dict, schemas: dict, credential_definitions: dict*)

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 schemas_json

delete_credential (*credential_id: str*)

Remove a credential stored in the wallet.

Parameters **credential_id** – Credential id to remove

get_credential (*credential_id: str*)

Get a credential stored in the wallet.

Parameters **credential_id** – Credential id to retrieve

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

Get credentials stored in the wallet.

Parameters

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

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

Get credentials stored in the wallet.

Parameters

- **presentation_request** – Valid presentation request from issuer
- **referents** – Presentation request referents to use to search for creds
- **start** – Starting index
- **count** – Maximum number of records to return
- **extra_query** – wql query dict

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

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

Parameters

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

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

store_credential (*credential_definition, credential_data, credential_request_metadata, credential_attr_mime_types=None, credential_id=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

1.1.6 aries_cloudagent.issuer package

Submodules

aries_cloudagent.issuer.base module

Ledger issuer class.

class `aries_cloudagent.issuer.base.BaseIssuer`

Bases: `abc.ABC`

Base class for issuer.

aries_cloudagent.issuer.indy module

Indy issuer implementation.

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

Bases: `aries_cloudagent.issuer.base.BaseIssuer`

Indy issuer class.

create_credential (*schema, credential_offer, credential_request, credential_values*)

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

Returns A tuple of created credential, revocation id

create_credential_offer (*credential_definition_id: str*)

Create a credential offer for the given credential definition id.

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

Returns A credential offer

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

Bases: `aries_cloudagent.core.error.BaseError`

Generic issuer error.

aries_cloudagent.issuer.util module

Issuer utils.

`aries_cloudagent.issuer.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

1.1.7 aries_cloudagent.ledger package

Submodules

aries_cloudagent.ledger.base module

Ledger base class.

class `aries_cloudagent.ledger.base.BaseLedger`

Bases: `abc.ABC`

Base class for ledger.

LEDGER_TYPE = None

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

did_to_nym (did: str) → str
Remove the ledger's DID prefix to produce a nym.

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

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

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

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

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

get_latest_txn_author_acceptance ()

Look up the latest TAA acceptance.

get_txn_author_agreement (*reload: bool = False*)

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

nym_to_did (*nym: str*) → str

Format a nym with the ledger's DID prefix.

register_nym (*did: str, verkey: str, alias: str = None, role: str = None*)

Register a nym on the ledger.

Parameters

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

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

Generate the digest of a TAA record.

update_endpoint_for_did (*did: str, endpoint: str*) → bool

Check and update the endpoint on the ledger.

Parameters

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

aries_cloudagent.ledger.error module

Ledger related errors.

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

Bases: `aries_cloudagent.ledger.error.LedgerError`

The current request cannot proceed.

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

Bases: `aries_cloudagent.ledger.error.LedgerError`

Indy pool is closed.

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

Bases: `aries_cloudagent.ledger.error.LedgerError`

Base class for ledger configuration errors.

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

Bases: `aries_cloudagent.core.error.BaseError`

Base class for ledger errors.

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

Bases: `aries_cloudagent.ledger.error.LedgerError`

The ledger rejected the transaction.

aries_cloudagent.ledger.indy module

Indy ledger implementation.

```
class aries_cloudagent.ledger.indy.IndyErrorHandler (message: str = None, error_cls:
                                                    Type[aries_cloudagent.ledger.error.LedgerError]
                                                    =
                                                    <class
                                                    'aries_cloudagent.ledger.error.LedgerError'>)
```

Bases: `object`

Trap IndyError and raise an appropriate LedgerError instead.

```
classmethod wrap_error (err_value: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538c3034e0>, message: str = None, error_cls:
                                Type[aries_cloudagent.ledger.error.LedgerError]
                                =
                                <class 'aries_cloudagent.ledger.error.LedgerError'>) →
                                aries_cloudagent.ledger.error.LedgerError
```

Create an instance of LedgerError from an IndyError.

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

Bases: `aries_cloudagent.ledger.base.BaseLedger`

Indy ledger class.

```
LEDGER_TYPE = 'indy'
```

```
accept_txn_author_agreement (taa_record: dict, mechanism: str, accept_time: int = None)
    Save a new record recording the acceptance of the TAA.
```

```
check_existing_schema (public_did: str, schema_name: str, schema_version: str, at-
                        tribute_names: Sequence[str]) → str
    Check if a schema has already been published.
```

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

```
close ()
    Close the pool ledger.
```

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

```
credential_definition_id2schema_id (credential_definition_id)
    From a credential definition, get the identifier for its schema.
```

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

```
fetch_credential_definition (credential_definition_id: str)
    Get a credential definition from the ledger by id.
```

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

fetch_schema_by_id (*schema_id: str*)

Get schema from ledger.

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

Returns Indy schema dict

fetch_schema_by_seq_no (*seq_no: int*)

Fetch a schema by its sequence number.

Parameters `seq_no` – schema ledger sequence number

Returns Indy schema dict

fetch_txn_author_agreement ()

Fetch the current AML and TAA from the ledger.

get_credential_definition (*credential_definition_id: str*)

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

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

get_endpoint_for_did (*did: str*) → str

Fetch the endpoint for a ledger DID.

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

get_indy_storage () → `aries_cloudagent.storage.indy.IndyStorage`

Get an IndyStorage instance for the current wallet.

get_key_for_did (*did: str*) → str

Fetch the verkey for a ledger DID.

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

get_latest_txn_author_acceptance ()

Look up the latest TAA acceptance.

get_schema (*schema_id: str*)

Get a schema from the cache if available, otherwise fetch from the ledger.

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

get_txn_author_agreement (*reload: bool = False*)

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

nym_to_did (*nym: str*) → str

Format a nym with the ledger's DID prefix.

open ()

Open the pool ledger, creating it if necessary.

register_nym (*did: str, verkey: str, alias: str = None, role: str = None*)

Register a nym on the ledger.

Parameters

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

send_credential_definition (*schema_id: str, tag: str = None*)

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

Parameters

- **schema_id** – The schema id of the schema to create cred def for
- **tag** – Option tag to distinguish multiple credential definitions

send_schema (*schema_name: str, schema_version: str, attribute_names: Sequence[str]*)

Send schema to ledger.

Parameters

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

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

Generate the digest of a TAA record.

taa_rough_timestamp () → int

Get a timestamp accurate to the day.

Anything more accurate is a privacy concern.

update_endpoint_for_did (*did: str, endpoint: str*) → bool

Check and update the endpoint on the ledger.

Parameters

- **did** – The ledger DID
- **endpoint** – The endpoint address
- **transport_vk** – The endpoint transport verkey

aries_cloudagent.ledger.provider module

Default ledger provider classes.

class aries_cloudagent.ledger.provider.LedgerProvider

Bases: *aries_cloudagent.config.base.BaseProvider*

Provider for the default ledger implementation.

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

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

1.1.8 aries_cloudagent.messaging package

Subpackages

aries_cloudagent.messaging.actionmenu package

Subpackages

`aries_cloudagent.messaging.actionmenu.handlers` package

Submodules

`aries_cloudagent.messaging.actionmenu.handlers.menu_handler` module

`aries_cloudagent.messaging.actionmenu.handlers.menu_request_handler` module

`aries_cloudagent.messaging.actionmenu.handlers.perform_handler` module

`aries_cloudagent.messaging.actionmenu.messages` package

Submodules

`aries_cloudagent.messaging.actionmenu.messages.menu` module

`aries_cloudagent.messaging.actionmenu.messages.menu_request` module

`aries_cloudagent.messaging.actionmenu.messages.perform` module

`aries_cloudagent.messaging.actionmenu.models` package

Submodules

`aries_cloudagent.messaging.actionmenu.models.menu_form` module

`aries_cloudagent.messaging.actionmenu.models.menu_form_param` module

`aries_cloudagent.messaging.actionmenu.models.menu_option` module

Submodules

`aries_cloudagent.messaging.actionmenu.base_service` module

`aries_cloudagent.messaging.actionmenu.controller` module

`aries_cloudagent.messaging.actionmenu.driver_service` module

`aries_cloudagent.messaging.actionmenu.message_types` module

`aries_cloudagent.messaging.actionmenu.routes` module

`aries_cloudagent.messaging.actionmenu.util` module

`aries_cloudagent.messaging.basicmessage` package

Subpackages

`aries_cloudagent.messaging.basicmessage.handlers` package

Submodules

`aries_cloudagent.messaging.basicmessage.handlers.basicmessage_handler` module

`aries_cloudagent.messaging.basicmessage.messages` package

Submodules

`aries_cloudagent.messaging.basicmessage.messages.basicmessage` module

Submodules

`aries_cloudagent.messaging.basicmessage.message_types` module

`aries_cloudagent.messaging.basicmessage.routes` module

`aries_cloudagent.messaging.connections` package

Subpackages

`aries_cloudagent.messaging.connections.handlers` package

Submodules

`aries_cloudagent.messaging.connections.handlers.connection_invitation_handler` module

`aries_cloudagent.messaging.connections.handlers.connection_request_handler` module

`aries_cloudagent.messaging.connections.handlers.connection_response_handler` module

`aries_cloudagent.messaging.connections.messages` package

Submodules

`aries_cloudagent.messaging.connections.messages.connection_invitation` module

`aries_cloudagent.messaging.connections.messages.connection_request` module

`aries_cloudagent.messaging.connections.messages.connection_response` module

`aries_cloudagent.messaging.connections.messages.problem_report` module

`aries_cloudagent.messaging.connections.models` package

Subpackages

`aries_cloudagent.messaging.connections.models.diddoc` package

Submodules

`aries_cloudagent.messaging.connections.models.diddoc.diddoc` module

`aries_cloudagent.messaging.connections.models.diddoc.publickey` module

`aries_cloudagent.messaging.connections.models.diddoc.service` module

`aries_cloudagent.messaging.connections.models.diddoc.util` module

Submodules

`aries_cloudagent.messaging.connections.models.connection_detail` module

`aries_cloudagent.messaging.connections.models.connection_record` module

`aries_cloudagent.messaging.connections.models.connection_target` module

Submodules

`aries_cloudagent.messaging.connections.manager` module

`aries_cloudagent.messaging.connections.message_types` module

`aries_cloudagent.messaging.connections.routes` module

`aries_cloudagent.messaging.credential_definitions` package

Submodules

`aries_cloudagent.messaging.credential_definitions.routes` module

Credential definition admin routes.


```
class aries_cloudagent.messaging.credential_definitions.routes.CredentialDefinitionGetResu
```

Bases: `marshmallow.schema.Schema`

Results schema for schema get request.

opts = `<marshmallow.schema.SchemaOpts object>`

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

Bases: `marshmallow.schema.Schema`

Credential definition schema.

opts = `<marshmallow.schema.SchemaOpts object>`

```
class aries_cloudagent.messaging.credential_definitions.routes.CredentialDefinitionSendReq
```

Bases: `marshmallow.schema.Schema`

Request schema for schema send request.

opts = `<marshmallow.schema.SchemaOpts object>`

```
class aries_cloudagent.messaging.credential_definitions.routes.CredentialDefinitionSendRes
```

Bases: `marshmallow.schema.Schema`

Results schema for schema send request.

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
class aries_cloudagent.messaging.credential_definitions.routes.CredentialDefinitionsCreated
```

Bases: `marshmallow.schema.Schema`

Results schema for cred-defs-created request.

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
aries_cloudagent.messaging.credential_definitions.routes.credential_definitions_created(req
```

Request handler for retrieving credential definitions that current agent created.

Parameters `request` – aiohttp request object

Returns The identifiers of matching credential definitions.

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

Request handler for getting a credential definition from the ledger.

Parameters `request` – aiohttp request object

Returns The credential definition details.

`aries_cloudagent.messaging.credential_definitions.routes.credential_definitions_send_creden`

Request handler for sending a credential definition to the ledger.

Parameters **request** – aiohttp request object

Returns The credential definition identifier

`aries_cloudagent.messaging.credential_definitions.routes.register` (*app:*
<sphinx.ext.autodoc.importer._Mock
object at
0x7f538bac5cc0>)

Register routes.

aries_cloudagent.messaging.credentials package

Subpackages

aries_cloudagent.messaging.credentials.handlers package

Submodules

aries_cloudagent.messaging.credentials.handlers.credential_issue_handler module

aries_cloudagent.messaging.credentials.handlers.credential_offer_handler module

aries_cloudagent.messaging.credentials.handlers.credential_request_handler module

aries_cloudagent.messaging.credentials.handlers.credential_stored_handler module

aries_cloudagent.messaging.credentials.messages package

Submodules

aries_cloudagent.messaging.credentials.messages.credential_issue module

aries_cloudagent.messaging.credentials.messages.credential_offer module

aries_cloudagent.messaging.credentials.messages.credential_request module

aries_cloudagent.messaging.credentials.messages.credential_stored module

aries_cloudagent.messaging.credentials.models package

Submodules

[aries_cloudagent.messaging.credentials.models.credential_exchange module](#)

Submodules

[aries_cloudagent.messaging.credentials.manager module](#)

[aries_cloudagent.messaging.credentials.message_types module](#)

[aries_cloudagent.messaging.credentials.routes module](#)

[aries_cloudagent.messaging.decorators package](#)

Submodules

[aries_cloudagent.messaging.decorators.base module](#)

Classes for managing a collection of decorators.

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

    Bases: collections.OrderedDict
    Collection of decorators.

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

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

    extract_decorators (message: Mapping[KT, VT_co], schema:
                        Type[<sphinx.ext.autodoc.importer._MockObject object at
                        0x7f538c6586d8>] = None, serialized: bool = True, skip_attrs: Se-
                        quence[str] = None) → collections.OrderedDict
        Extract decorators and return the remaining properties.

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

    fields
        Accessor for the set of currently defined fields.

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

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

    models
        Accessor for the models dictionary.

    prefix
        Accessor for the decorator prefix.
```

remove_field (*name: str*)

Remove a named decorated field.

remove_model (*key: str*)

Remove a registered decorator model.

to_dict (*prefix: str = None*) → collections.OrderedDict

Convert to a dictionary (serialize).

Raises BaseModelError – on decorator validation errors

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

Bases: aries_cloudagent.core.error.BaseError

Base error for decorator issues.

aries_cloudagent.messaging.decorators.default module

Default decorator set implementation.

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

Bases: aries_cloudagent.messaging.decorators.base.BaseDecoratorSet

Default decorator set implementation.

aries_cloudagent.messaging.decorators.localization_decorator module

The localization decorator (~110n) for message localization information.

```
class aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecorator (*,
```

Bases: aries_cloudagent.messaging.models.base.BaseModel

Class representing the localization decorator.

```
class Meta
```

```
Bases: object
```

```
LocalizationDecorator metadata.
```

```
schema_class = 'LocalizationDecoratorSchema'
```

```
class aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecoratorSchema
```

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

```
Localization decorator schema used in serialization/deserialization.
```

```
class Meta
```

```
Bases: object
```

```
LocalizationDecoratorSchema metadata.
```

```
model_class
```

```
alias of LocalizationDecorator
```

```
catalogs
```

```
Used by autodoc_mock_imports.
```

```
locale
```

```
Used by autodoc_mock_imports.
```

```
localizable
```

```
Used by autodoc_mock_imports.
```

aries_cloudagent.messaging.decorators.signature_decorator module

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

```
class aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator(*,
                                                                                   sig-
                                                                                   na-
                                                                                   ture_type:
                                                                                   str
                                                                                   =
                                                                                   None,
                                                                                   sig-
                                                                                   na-
                                                                                   ture:
                                                                                   str
                                                                                   =
                                                                                   None,
                                                                                   sig_data:
                                                                                   str
                                                                                   =
                                                                                   None,
                                                                                   signer:
                                                                                   str
                                                                                   =
                                                                                   None)
```

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

```
Class representing a field value signed by a known verkey.
```

```

class Meta
    Bases: object

    SignatureDecorator metadata.

    schema_class = 'SignatureDecoratorSchema'

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

classmethod create (value, signer: str, wallet: aries_cloudagent.wallet.base.BaseWallet, times-
                    tamp=None) → aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator
    Create a Signature.

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

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

    Returns The created SignatureDecorator object

decode () -> (<class 'object'>, <class 'int'>)
    Decode the signature to its timestamp and value.

    Returns A tuple of (decoded message, timestamp)

verify (wallet: aries_cloudagent.wallet.base.BaseWallet) → bool
    Verify the signature against the signer's public key.

    Parameters wallet – Wallet to use to verify signature

    Returns True if verification succeeds else False

class aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecoratorSchema (*a, **kw)
    Bases: aries_cloudagent.messaging.models.base.BaseModelSchema

    SignatureDecorator schema.

    class Meta
        Bases: object

        SignatureDecoratorSchema metadata.

        model_class
            alias of SignatureDecorator

    sig_data
        Used by autodoc_mock_imports.

    signature
        Used by autodoc_mock_imports.

    signature_type
        Used by autodoc_mock_imports.

    signer
        Used by autodoc_mock_imports.

```

aries_cloudagent.messaging.decorators.thread_decorator module

A message decorator for threads.

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

```
class aries_cloudagent.messaging.decorators.thread_decorator.ThreadDecorator (*,  
    thid:  
    str  
    =  
    None,  
    pthid:  
    str  
    =  
    None,  
    sender_order:  
    int  
    =  
    None,  
    re-  
    ceived_orders:  
    Map-  
    ping[KT,  
    VT_co]  
    =  
    None)
```

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

Class representing thread decorator.

class Meta

Bases: *object*

ThreadDecorator metadata.

schema_class = 'ThreadDecoratorSchema'

pthid

Accessor for parent thread identifier.

Returns This thread's *pthid*

received_orders

Get received orders.

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

sender_order

Get sender order.

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

thid

Accessor for thread identifier.

Returns This thread's *thid*

```
class aries_cloudagent.messaging.decorators.thread_decorator.ThreadDecoratorSchema (*args,  
    **kwargs)  
Bases: aries_cloudagent.messaging.models.base.BaseModelSchema
```


Thread decorator schema used in serialization/deserialization.

class Meta

Bases: `object`

ThreadDecoratorSchema metadata.

model_class

alias of `ThreadDecorator`

pthid

Used by `autodoc_mock_imports`.

received_orders

Used by `autodoc_mock_imports`.

sender_order

Used by `autodoc_mock_imports`.

thid

Used by `autodoc_mock_imports`.

aries_cloudagent.messaging.decorators.timing_decorator module

The timing decorator (~timing).

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

```

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

```

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

Class representing the timing decorator.

```

class Meta
    Bases: object
    TimingDecorator metadata.
    schema_class = 'TimingDecoratorSchema'

```

```

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

```

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

Timing decorator schema used in serialization/deserialization.

```

class Meta
    Bases: object
    TimingDecoratorSchema metadata.

```

model_class
alias of *TimingDecorator*

delay_milli
Used by autodoc_mock_imports.

expires_time
Used by autodoc_mock_imports.

in_time
Used by autodoc_mock_imports.

out_time
Used by autodoc_mock_imports.

stale_time
Used by autodoc_mock_imports.

wait_until_time
Used by autodoc_mock_imports.

aries_cloudagent.messaging.decorators.transport_decorator module

The transport decorator (~transport).

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

```
class aries_cloudagent.messaging.decorators.transport_decorator.TransportDecorator (*,
re-
turn_route:
str
=
None,
re-
turn_route:
str
=
None,
queued_me
int
=
None)
```

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

Class representing the transport decorator.

```
class Meta
Bases: object

TransportDecorator metadata.

schema_class = 'TransportDecoratorSchema'
```

```
class aries_cloudagent.messaging.decorators.transport_decorator.TransportDecoratorSchema (*,
**)
```

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

Transport decorator schema used in serialization/deserialization.

```
class Meta
Bases: object
```

TransportDecoratorSchema metadata.

model_class

alias of *TransportDecorator*

queued_message_count

Used by autodoc_mock_imports.

return_route

Used by autodoc_mock_imports.

return_route_thread

Used by autodoc_mock_imports.

aries_cloudagent.messaging.discovery package

Subpackages

aries_cloudagent.messaging.discovery.handlers package

Submodules

aries_cloudagent.messaging.discovery.handlers.disclose_handler module

aries_cloudagent.messaging.discovery.handlers.query_handler module

aries_cloudagent.messaging.discovery.messages package

Submodules

aries_cloudagent.messaging.discovery.messages.disclose module

aries_cloudagent.messaging.discovery.messages.query module

Submodules

aries_cloudagent.messaging.discovery.message_types module

aries_cloudagent.messaging.discovery.routes module

aries_cloudagent.messaging.introduction package

Subpackages

aries_cloudagent.messaging.introduction.handlers package

Submodules

`aries_cloudagent.messaging.introduction.handlers.forward_invitation_handler` module

`aries_cloudagent.messaging.introduction.handlers.invitation_handler` module

`aries_cloudagent.messaging.introduction.handlers.invitation_request_handler` module

`aries_cloudagent.messaging.introduction.messages` package

Submodules

`aries_cloudagent.messaging.introduction.messages.forward_invitation` module

`aries_cloudagent.messaging.introduction.messages.invitation` module

`aries_cloudagent.messaging.introduction.messages.invitation_request` module

Submodules

`aries_cloudagent.messaging.introduction.base_service` module

`aries_cloudagent.messaging.introduction.demo_service` module

`aries_cloudagent.messaging.introduction.message_types` module

`aries_cloudagent.messaging.introduction.routes` module

`aries_cloudagent.messaging.models` package

Submodules

`aries_cloudagent.messaging.models.base` module

Base classes for Models and Schemas.

```
class aries_cloudagent.messaging.models.base.BaseModel
```

Bases: `abc.ABC`

Base model that provides convenience methods.

```
class Meta
```

Bases: `object`

BaseModel meta data.

```
schema_class = None
```

Schema

Accessor for the model's schema class.

Returns The schema class

```
classmethod deserialize(obj)
```

Convert from JSON representation to a model instance.

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

Returns A model instance for this data

classmethod `from_json(json_repr: Union[str, bytes])`

Parse a JSON string into a model instance.

Parameters `json_repr` – JSON string

Returns A model instance representation of this JSON

serialize (`as_string=False`) → dict

Create a JSON-compatible dict representation of the model instance.

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

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

to_json () → str

Create a JSON representation of the model instance.

Returns A JSON representation of this message

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

Bases: `aries_cloudagent.core.error.BaseError`

Base exception class for base model errors.

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

Bases: `sphinx.ext.autodoc.importer._MockObject`

BaseModel schema.

class `Meta`

Bases: `object`

BaseModelSchema metadata.

model_class = None

ordered = True

skip_values = [None]

Model

Accessor for the schema's model class.

Returns The model class

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

Return model instance after loading.

Returns A model instance

remove_skipped_values (`data, **kwargs`)

Remove values that are are marked to skip.

Returns Returns this modified data

skip_dump_only (`data, **kwargs`)

Skip fields that are only expected during serialization.

Parameters `data` – The incoming data to clean

Returns The modified data

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

Resolve a class.

Parameters

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

Returns The resolved class

Raises `ClassNotFoundError` – If the class could not be loaded

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

Resolve a meta property.

Parameters

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

Returns The meta property

`aries_cloudagent.messaging.models.base_record` module

Classes for BaseStorage-based record management.

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

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

Represents a single storage record.

CACHE_ENABLED = **False**

CACHE_TTL = **60**

LOG_STATE_FLAG = **None**

class Meta

Bases: `object`

BaseRecord metadata.

RECORD_ID_NAME = **'id'**

RECORD_TYPE = **None**

TAG_NAMES = **{'state'}**

WEBHOOK_TOPIC = **None**

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

Assemble a cache key.

Parameters

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

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

Clear the cached value of this record, if any.

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

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

Parameters

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

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

Remove the stored record.

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

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

Initialize a record from its stored representation.

Parameters

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

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

Shortcut method to fetch a cached key value.

Parameters

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

classmethod **get_tag_map** () → Mapping[str, str]

Accessor for the set of defined tags.

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

Print a message with increased visibility (for testing).

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

Perform post-save actions.

Parameters

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

classmethod `prefix_tag_filter` (*tag_filter: dict*)

Prefix unencrypted tags used in the tag filter.

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

Query stored records.

Parameters

- **context** – The injection context to use
- **tag_filter** – An optional dictionary of tag filter clauses
- **post_filter** – Additional value filters to apply

record_tags

Accessor to define implementation-specific tags.

record_value

Accessor to define custom properties for the JSON record value.

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

Retrieve a stored record by ID.

Parameters

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

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

Retrieve a record by tag filter.

Parameters

- **context** – The injection context to use
- **tag_filter** – The filter dictionary to apply
- **post_filter** – Additional value filters to apply after retrieval

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

Persist the record to storage.

Parameters

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

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

Send a standard webhook.

Parameters

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

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

Shortcut method to set a cached key value.

Parameters

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

storage_record

Accessor for a *StorageRecord* representing this record.

classmethod strip_tag_prefix (*tags: dict*)

Strip tilde from unencrypted tag names.

tags

Accessor for the record tags generated for this record.

value

Accessor for the JSON record value generated for this record.

webhook_payload

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

webhook_topic

Return the webhook topic value.

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

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

Schema to allow serialization/deserialization of base records.

class Meta

Bases: *object*

BaseRecordSchema metadata.

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

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

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

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

Determine if a record value matches the post-filter.

`aries_cloudagent.messaging.presentations` package

Subpackages

`aries_cloudagent.messaging.presentations.handlers` package

Submodules

`aries_cloudagent.messaging.presentations.handlers.credential_presentation_handler` module

`aries_cloudagent.messaging.presentations.handlers.presentation_request_handler` module

`aries_cloudagent.messaging.presentations.messages` package

Submodules

`aries_cloudagent.messaging.presentations.messages.credential_presentation` module

`aries_cloudagent.messaging.presentations.messages.presentation_request` module

`aries_cloudagent.messaging.presentations.models` package

Submodules

`aries_cloudagent.messaging.presentations.models.presentation_exchange` module

Submodules

`aries_cloudagent.messaging.presentations.manager` module

`aries_cloudagent.messaging.presentations.message_types` module

`aries_cloudagent.messaging.presentations.routes` module

`aries_cloudagent.messaging.problem_report` package

Submodules

`aries_cloudagent.messaging.problem_report.handler` module

`aries_cloudagent.messaging.problem_report.message` module

`aries_cloudagent.messaging.routing` package

Subpackages

`aries_cloudagent.messaging.routing.handlers` package

Submodules

`aries_cloudagent.messaging.routing.handlers.forward_handler` module

`aries_cloudagent.messaging.routing.handlers.route_query_request_handler` module

`aries_cloudagent.messaging.routing.handlers.route_query_response_handler` module

`aries_cloudagent.messaging.routing.handlers.route_update_request_handler` module

`aries_cloudagent.messaging.routing.handlers.route_update_response_handler` module

`aries_cloudagent.messaging.routing.messages` package

Submodules

`aries_cloudagent.messaging.routing.messages.forward` module

`aries_cloudagent.messaging.routing.messages.route_query_request` module

`aries_cloudagent.messaging.routing.messages.route_query_response` module

`aries_cloudagent.messaging.routing.messages.route_update_request` module

`aries_cloudagent.messaging.routing.messages.route_update_response` module

`aries_cloudagent.messaging.routing.models` package

Submodules

`aries_cloudagent.messaging.routing.models.paginate` module

`aries_cloudagent.messaging.routing.models.paginated` module

`aries_cloudagent.messaging.routing.models.route_query_result` module

`aries_cloudagent.messaging.routing.models.route_record` module

`aries_cloudagent.messaging.routing.models.route_update` module

`aries_cloudagent.messaging.routing.models.route_updated` module

Submodules

[aries_cloudagent.messaging.routing.manager module](#)

[aries_cloudagent.messaging.routing.message_types module](#)

[aries_cloudagent.messaging.schemas package](#)

Submodules

[aries_cloudagent.messaging.schemas.routes module](#)

Credential schema admin routes.

```
class aries_cloudagent.messaging.schemas.routes.SchemaGetResultsSchema (*,
                                                                    only=None,
                                                                    ex-
                                                                    clude=(),
                                                                    many=False,
                                                                    con-
                                                                    text=None,
                                                                    load_only=(),
                                                                    dump_only=(),
                                                                    par-
                                                                    tial=False,
                                                                    un-
                                                                    known=None)
```

Bases: `marshmallow.schema.Schema`

Results schema for schema get request.

opts = `<marshmallow.schema.SchemaOpts object>`

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

Bases: `marshmallow.schema.Schema`

Content for returned schema.

opts = `<marshmallow.schema.SchemaOpts object>`

```

class aries_cloudagent.messaging.schemas.routes.SchemaSendRequestSchema (*,
                                                                    only=None,
                                                                    ex-
                                                                    clude=(),
                                                                    many=False,
                                                                    con-
                                                                    text=None,
                                                                    load_only=(),
                                                                    dump_only=(),
                                                                    par-
                                                                    tial=False,
                                                                    un-
                                                                    known=None)

    Bases: marshmallow.schema.Schema

    Request schema for schema send request.

    opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.schemas.routes.SchemaSendResultsSchema (*,
                                                                    only=None,
                                                                    ex-
                                                                    clude=(),
                                                                    many=False,
                                                                    con-
                                                                    text=None,
                                                                    load_only=(),
                                                                    dump_only=(),
                                                                    par-
                                                                    tial=False,
                                                                    un-
                                                                    known=None)

    Bases: marshmallow.schema.Schema

    Results schema for schema send request.

    opts = <marshmallow.schema.SchemaOpts object>

class aries_cloudagent.messaging.schemas.routes.SchemasCreatedResultsSchema (*,
                                                                    only=None,
                                                                    ex-
                                                                    clude=(),
                                                                    many=False,
                                                                    con-
                                                                    text=None,
                                                                    load_only=(),
                                                                    dump_only=(),
                                                                    par-
                                                                    tial=False,
                                                                    un-
                                                                    known=None)

    Bases: marshmallow.schema.Schema

    Results schema for a schemas-created request.

    opts = <marshmallow.schema.SchemaOpts object>

```

```
aries_cloudagent.messaging.schemas.routes.register(app:
    <sphinx.ext.autodoc.importer._MockObject
    object at 0x7f538bb7d7b8>)
```

Register routes.

```
aries_cloudagent.messaging.schemas.routes.schemas_created(request:
    <sphinx.ext.autodoc.importer._MockObject
    object at
    0x7f538bb7d7b8>)
```

Request handler for retrieving schemas that current agent created.

Parameters **request** – aiohttp request object

Returns The identifiers of matching schemas

```
aries_cloudagent.messaging.schemas.routes.schemas_get_schema(request:
    <sphinx.ext.autodoc.importer._MockObject
    object at
    0x7f538bb7d7b8>)
```

Request handler for sending a credential offer.

Parameters **request** – aiohttp request object

Returns The schema details.

```
aries_cloudagent.messaging.schemas.routes.schemas_send_schema(request:
    <sphinx.ext.autodoc.importer._MockObject
    object at
    0x7f538bb7d7b8>)
```

Request handler for sending a credential offer.

Parameters **request** – aiohttp request object

Returns The schema id sent

aries_cloudagent.messaging.trustping package

Subpackages

aries_cloudagent.messaging.trustping.handlers package

Submodules

aries_cloudagent.messaging.trustping.handlers.ping_handler module

aries_cloudagent.messaging.trustping.handlers.ping_response_handler module

aries_cloudagent.messaging.trustping.messages package

Submodules

aries_cloudagent.messaging.trustping.messages.ping module

aries_cloudagent.messaging.trustping.messages.ping_response module

Submodules

[aries_cloudagent.messaging.trustping.message_types module](#)

[aries_cloudagent.messaging.trustping.routes module](#)

Submodules

[aries_cloudagent.messaging.agent_message module](#)

Agent message base class and schema.

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

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

Agent message base class.

Handler

Accessor for the agent message's handler class.

Returns Handler class

class Meta

Bases: *object*

AgentMessage metadata.

handler_class = None

message_type = None

schema_class = None

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

Copy thread information from a previous message.

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

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

Assign a specific thread ID.

Parameters

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

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

Get the signature for a named field.

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

Returns A SignatureDecorator for the requested field name

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

Add or replace the signature for a named field.

Parameters

- **field_name** – Field to set signature on

- **signature** – Signature for the field

sign_field (*field_name: str, signer_verkey: str, wallet: aries_cloudagent.wallet.base.BaseWallet, timestamp=None*) → *aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator*
Create and store a signature for a named field.

Parameters

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

Returns A SignatureDecorator for newly created signature

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

verify_signatures (*wallet: aries_cloudagent.wallet.base.BaseWallet*) → `bool`
Verify all associated field signatures.

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

Returns True if all signatures verify, else false

verify_signed_field (*field_name: str, wallet: aries_cloudagent.wallet.base.BaseWallet, signer_verkey: str = None*) → `str`
Verify a specific field signature.

Parameters

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

Returns The verkey of the signer

Raises

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

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

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

Base exception for agent message issues.

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

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

AgentMessage schema.

```
class Meta
    Bases: object

    AgentMessageSchema metadata.

    model_class = None

    signed_fields = None

    check_dump_decorators (obj, **kwargs)
        Pre-dump hook to validate and load the message decorators.

        Parameters obj – The AgentMessage object

        Raises BaseModelError – If a decorator does not validate

    dump_decorators (data, **kwargs)
        Post-dump hook to write the decorators to the serialized output.

        Parameters obj – The serialized data

        Returns The modified data

    extract_decorators (data, **kwargs)
        Pre-load hook to extract the decorators and check the signed fields.

        Parameters data – Incoming data to parse

        Returns Parsed and modified data

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

    populate_decorators (obj, **kwargs)
        Post-load hook to populate decorators on the message.

        Parameters obj – The AgentMessage object

        Returns The AgentMessage object with populated decorators

    replace_signatures (data, **kwargs)
        Post-dump hook to write the signatures to the serialized output.

        Parameters obj – The serialized data

        Returns The modified data
```

aries_cloudagent.messaging.base_context module

aries_cloudagent.messaging.base_handler module

A Base handler class for all message handlers.

```
class aries_cloudagent.messaging.base_handler.BaseHandler
    Bases: abc.ABC

    Abstract base class for handlers.
```

handle (*context:* *aries_cloudagent.messaging.request_context.RequestContext*, *responder:* *aries_cloudagent.messaging.responder.BaseResponder*)
 Abstract method for handler logic.

Parameters

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

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

Bases: *aries_cloudagent.core.error.BaseError*

Exception base class for generic handler errors.

aries_cloudagent.messaging.error module

Messaging-related error classes and codes.

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

Bases: *aries_cloudagent.core.error.BaseError*

Message parse error.

error_code = 'message_parse_error'

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

Bases: *aries_cloudagent.core.error.BaseError*

Message preparation error.

error_code = 'message_prepare_error'

aries_cloudagent.messaging.message_delivery module

aries_cloudagent.messaging.outbound_message module

aries_cloudagent.messaging.protocol_registry module

aries_cloudagent.messaging.request_context module

Request context class.

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

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

Bases: `aries_cloudagent.config.injection_context.InjectionContext`

Context established by the Conductor and passed into message handlers.

connection_ready

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

Returns True if the connection is active, else False

connection_record

Accessor for the related connection record.

copy () → `aries_cloudagent.messaging.request_context.RequestContext`

Produce a copy of the request context instance.

default_endpoint

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

Returns The default agent endpoint

default_label

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

Returns The default label

message

Accessor for the deserialized message instance.

Returns This context's agent message

message_receipt

Accessor for the message receipt information.

Returns This context's message receipt information

aries_cloudagent.messaging.responder module

A message responder.

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

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

Bases: `abc.ABC`

Interface for message handlers to send responses.

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

Create an *OutboundMessage* from a message payload.

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

Convert a message to an *OutboundMessage* and send it.

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

Send an outbound message.

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

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

Send a reply to an incoming message.

Parameters

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

Raises *ResponderError* – If there is no active connection

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

Dispatch a webhook.

Parameters

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

class *aries_cloudagent.messaging.responder.MockResponder*

Bases: *aries_cloudagent.messaging.responder.BaseResponder*

Mock responder implementation for use by tests.

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

Convert a message to an *OutboundMessage* and send it.

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

Send an outbound message.

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

Send a reply to an incoming message.

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

Send an outbound message.

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

Bases: `aries_cloudagent.core.error.BaseError`

Responder error.

`aries_cloudagent.messaging.serializer` module

`aries_cloudagent.messaging.socket` 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.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.

1.1.9 `aries_cloudagent.storage` package

Submodules

`aries_cloudagent.storage.base` module

Abstract base classes for non-secrets storage.

class aries_cloudagent.storage.base.BaseStorage

Bases: `abc.ABC`

Abstract Non-Secrets interface.

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

Add a new record to the store.

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

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

Delete an existing record.

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

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

Update an existing stored record's tags.

Parameters

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

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

Fetch a record from the store by type and ID.

Parameters

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

Returns A *StorageRecord* instance

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

Create a new record query.

Parameters

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

Returns An instance of *BaseStorageRecordSearch*

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

Update an existing stored record's tags.

Parameters

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

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

Update an existing stored record's value.

Parameters

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

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

Bases: `abc.ABC`

Represent an active stored records search.

close()

Dispose of the search query.

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

Fetch the next list of results from the store.

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

Returns A list of *StorageRecord*

fetch_all () → Sequence[aries_cloudagent.storage.record.StorageRecord]

Fetch all records from the query.

fetch_single () → aries_cloudagent.storage.record.StorageRecord

Fetch a single query result.

handle

Handle a search request.

open ()

Start the search query.

opened

Accessor for open state.

Returns True if opened, else False

option (*name: str, default=None*)

Fetch a named search option, if defined.

Returns The option value or default

options

Accessor for the search options.

Returns The search options

page_size

Accessor for page size.

Returns The page size

store

BaseStorage backend for this implementation.

Returns The *BaseStorage* implementation being used

tag_query

Accessor for tag query.

Returns The tag query

type_filter

Accessor for type filter.

Returns The type filter

aries_cloudagent.storage.basic module

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

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

Bases: *aries_cloudagent.storage.base.BaseStorage*

Basic in-memory storage class.

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

Add a new record to the store.

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

Raises

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

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

Delete a record.

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

Raises *StorageNotFoundError* – If record not found

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

Update an existing stored record's tags.

Parameters

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

Raises *StorageNotFoundError* – If record not found

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

Fetch a record from the store by type and ID.

Parameters

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

Returns A *StorageRecord* instance

Raises *StorageNotFoundError* – If the record is not found

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

Parameters

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

Returns An instance of *BaseStorageRecordSearch*

update_record_tags (*record*: *aries_cloudagent.storage.record.StorageRecord*, *tags*: *Mapping[KT, VT_co]*)
Update an existing stored record's tags.

Parameters

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

Raises *StorageNotFoundError* – If record not found

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

Parameters

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

Raises *StorageNotFoundError* – If record not found

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

Bases: *aries_cloudagent.storage.base.BaseStorageRecordSearch*

Represent an active stored records search.

close ()

Dispose of the search query.

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

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

Returns A list of *StorageRecord*

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

open ()

Start the search query.

opened

Accessor for open state.

Returns True if opened, else False

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

Match simple tag filters (string values).

`aries_cloudagent.storage.basic.basic_tag_value_match(value: str, match: dict) → bool`

Match a single tag against a tag subquery.

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

aries_cloudagent.storage.error module

Storage-related exceptions.

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

Bases: `aries_cloudagent.storage.error.StorageError`

Duplicate record found in storage.

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

Bases: `aries_cloudagent.core.error.BaseError`

Base class for Storage errors.

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

Bases: `aries_cloudagent.storage.error.StorageError`

Record not found in storage.

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

Bases: `aries_cloudagent.storage.error.StorageError`

General exception during record search.

aries_cloudagent.storage.indy module

Indy implementation of BaseStorage interface.

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

Bases: `aries_cloudagent.storage.base.BaseStorage`

Indy Non-Secrets interface.

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

Add a new record to the store.

Parameters `record` – `StorageRecord` to be stored

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

Delete a record.

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

Raises

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

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

Update an existing stored record's tags.

Parameters

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

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

Fetch a record from the store by type and ID.

Parameters

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

Returns A *StorageRecord* instance

Raises

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

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

Search stored records.

Parameters

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

Returns An instance of *IndyStorageRecordSearch*

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

Update an existing stored record's tags.

Parameters

- **record** – *StorageRecord* to update

- **tags** – New tags

Raises

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

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

Update an existing stored record's value.

Parameters

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

Raises

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

wallet

Accessor for IndyWallet instance.

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

Bases: `aries_cloudagent.storage.base.BaseStorageRecordSearch`

Represent an active stored records search.

close()

Dispose of the search query.

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

Fetch the next list of results from the store.

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

Returns A list of *StorageRecord*

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

handle

Accessor for search handle.

Returns The handle

open()

Start the search query.

opened

Accessor for open state.

Returns True if opened, else False

aries_cloudagent.storage.provider module

Default storage provider classes.

class aries_cloudagent.storage.provider.StorageProvider

Bases: *aries_cloudagent.config.base.BaseProvider*

Provider for the default configurable storage classes.

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

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

aries_cloudagent.config.base.BaseInjector)

Create and return the storage instance.

aries_cloudagent.storage.record module

Record instance stored and searchable by BaseStorage implementation.

class aries_cloudagent.storage.record.StorageRecord

Bases: *aries_cloudagent.storage.record.StorageRecord*

Storage record class.

1.1.10 aries_cloudagent.tests package

Submodules

aries_cloudagent.tests.test_conductor module

aries_cloudagent.tests.test_dispatcher module

aries_cloudagent.tests.test_stats module

aries_cloudagent.tests.test_task_processor module

1.1.11 aries_cloudagent.transport package

Subpackages

aries_cloudagent.transport.inbound package

Submodules

aries_cloudagent.transport.inbound.base module

Base inbound transport class.

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

Bases: `abc.ABC`

Base inbound transport class.

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

Parameters

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

max_message_size

Accessor for this transport's max message size.

scheme

Accessor for this transport's scheme.

```
start () → None
```

Start listening for on this transport.

```
stop () → None
```

Stop listening for on this transport.

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

Bases: `tuple`

host

Alias for field number 1

module

Alias for field number 0

port

Alias for field number 2

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

Bases: `aries_cloudagent.transport.error.TransportError`

Generic inbound transport error.

```
exception aries_cloudagent.transport.inbound.base.InboundTransportRegistrationError (*args,
                                                                                      er-
                                                                                      ror_code:
                                                                                      str
                                                                                      =
                                                                                      None,
                                                                                      **kwargs)
```

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

Error in loading an inbound transport.

```
exception aries_cloudagent.transport.inbound.base.InboundTransportSetupError (*args,
                                                                                   er-
                                                                                   ror_code:
                                                                                   str
                                                                                   =
                                                                                   None,
                                                                                   **kwargs)
```

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

Setup error for an inbound transport.

aries_cloudagent.transport.inbound.http module

Http Transport classes and functions.

```
class aries_cloudagent.transport.inbound.http.HttpTransport (host: str, port:
                                                                int, create_session,
                                                                **kwargs)
```

Bases: `aries_cloudagent.transport.inbound.base.BaseInboundTransport`

Http Transport class.

```
inbound_message_handler (request: <sphinx.ext.autodoc.importer._MockObject object at
                                   0x7f538b693908>)
```

Message handler for inbound messages.

Parameters `request` – aiohttp request object

Returns The web response

```
invite_message_handler (request: <sphinx.ext.autodoc.importer._MockObject object at
                                   0x7f538b693908>)
```

Message handler for invites.

Parameters `request` – aiohttp request object

Returns The web response

```
make_application () → <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b693908>
Construct the aiohttp application.
```

```
start () → None
```

Start this transport.

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

```
stop () → None
```

Stop this transport.

aries_cloudagent.transport.inbound.manager module

Inbound transport manager.

```
class aries_cloudagent.transport.inbound.manager.InboundTransportManager (context:
    aries_cloudagent.config.inbound_manager.InboundManagerContext,
    re-ceive_inbound: Callable[[T_co, T_contra, V_co], Awaitable[T_co]],
    re-turn_inbound: Callable[[T_co, T_contra, V_co], Awaitable[T_co]])
```

Bases: `object`

Inbound transport manager class.

closed_session (session: *aries_cloudagent.transport.inbound.session.InboundSession*)
Clean up a closed session.

Returns an undelivered message to the caller if possible.

create_session (transport_type: *str*, *, accept_undelivered: *bool* = *False*,
can_respond: *bool* = *False*, client_info: *dict* = *None*, wire_format:
aries_cloudagent.transport.wire_format.BaseWireFormat = *None*)
Create a new inbound session.

Parameters

- **transport_type** – The inbound transport identifier
- **accept_undelivered** – Flag for accepting undelivered messages
- **can_respond** – Flag indicating that the transport can send responses
- **client_info** – An optional dict describing the client
- **wire_format** – Override the wire format for this session

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

get_transport_instance (transport_id: *str*) → *aries_cloudagent.transport.inbound.base.BaseInboundTransport*
Get an instance of a running transport by ID.

process_undelivered (session: *aries_cloudagent.transport.inbound.session.InboundSession*)
Interact with undelivered queue to find applicable messages.

Parameters session – The inbound session

register (config: *aries_cloudagent.transport.inbound.base.InboundTransportConfiguration*) → *str*
Register transport module.

Parameters config – The inbound transport configuration

register_transport (transport: *aries_cloudagent.transport.inbound.base.BaseInboundTransport*,
transport_id: *str*) → *str*
Register a new inbound transport class.

Parameters

- **transport** – Transport instance to register
- **transport_id** – The transport ID to register

return_to_session (*outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*)
→ bool

Return an outbound message via an open session, if possible.

return_undelivered (*outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*)
→ bool

Add an undelivered message to the undelivered queue.

At this point the message could not be associated with an inbound session and could not be delivered via an outbound transport.

setup ()
Perform setup operations.

start ()
Start all registered transports.

start_transport (*transport_id: str*)
Start a registered inbound transport.

Parameters **transport_id** – ID for the inbound transport to start

stop (*wait: bool = True*)
Stop all registered transports.

aries_cloudagent.transport.inbound.ws module

Websockets Transport classes and functions.

class `aries_cloudagent.transport.inbound.ws.WsTransport` (*host: str, port: int, create_session, **kwargs*)

Bases: `aries_cloudagent.transport.inbound.base.BaseInboundTransport`

Websockets Transport class.

inbound_message_handler (*request*)
Message handler for inbound messages.

Parameters **request** – aiohttp request object

Returns The web response

make_application () → `<sphinx.ext.autodoc.importer._MockObject object at 0x7f538c466978>`
Construct the aiohttp application.

scheme
Accessor for this transport's scheme.

start () → None
Start this transport.

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

stop () → None
Stop this transport.

aries_cloudagent.transport.outbound package

Subpackages

aries_cloudagent.transport.outbound.queue package

Submodules

aries_cloudagent.transport.outbound.queue.base module

aries_cloudagent.transport.outbound.queue.basic module

Submodules

aries_cloudagent.transport.outbound.base module

Base outbound transport.

```
class aries_cloudagent.transport.outbound.base.BaseOutboundTransport (wire_format:  
                                                                    aries_cloudagent.transport.wire  
                                                                    =  
                                                                    None)
```

Bases: `abc.ABC`

Base outbound transport class.

collector

Accessor for the stats collector instance.

handle_message (payload: *Union[str, bytes]*, endpoint: *str*)
Handle message from queue.

Parameters

- **payload** – message payload in string or byte format
- **endpoint** – URI endpoint for delivery

start ()
Start the transport.

stop ()
Shut down the transport.

wire_format
Accessor for a custom wire format for the transport.

```
exception aries_cloudagent.transport.outbound.base.OutboundDeliveryError (*args,  
                                                                           er-  
                                                                           ror_code:  
                                                                           str  
                                                                           =  
                                                                           None,  
                                                                           **kwargs)
```

Bases: `aries_cloudagent.transport.outbound.base.OutboundTransportError`

Base exception when a message cannot be delivered via an outbound transport.

```
exception aries_cloudagent.transport.outbound.base.OutboundTransportError (*args,
                                                                           er-
                                                                           ror_code:
                                                                           str
                                                                           =
                                                                           None,
                                                                           **kwargs)
```

Bases: `aries_cloudagent.transport.error.TransportError`

Generic outbound transport error.

```
exception aries_cloudagent.transport.outbound.base.OutboundTransportRegistrationError (*args,
                                                                                       er-
                                                                                       ror_co
                                                                                       str
                                                                                       =
                                                                                       None,
                                                                                       **kwa
```

Bases: `aries_cloudagent.transport.outbound.base.OutboundTransportError`

Outbound transport registration error.

aries_cloudagent.transport.outbound.http module

Http outbound transport.

```
class aries_cloudagent.transport.outbound.http.HttpTransport
    Bases: aries_cloudagent.transport.outbound.base.BaseOutboundTransport
    Http outbound transport class.
    handle_message (payload: Union[str, bytes], endpoint: str)
        Handle message from queue.
        Parameters message – OutboundMessage to send over transport implementation
    schemes = ('http', 'https')
    start ()
        Start the transport.
    stop ()
        Stop the transport.
```

aries_cloudagent.transport.outbound.manager module

Outbound transport manager.

```
class aries_cloudagent.transport.outbound.manager.OutboundTransportManager (context:
                                                                           aries_cloudagent.conf
                                                                           han-
                                                                           dle_not_delivered:
                                                                           Callable
                                                                           =
                                                                           None)
```

Bases: `object`

Outbound transport manager class.

deliver_queued_message (*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage*)
 → *_asyncio.Task*
 Kick off delivery of a queued message.

encode_queued_message (*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage*)
 → *_asyncio.Task*
 Kick off encoding of a queued message.

enqueue_message (*context: aries_cloudagent.config.injection_context.InjectionContext, outbound: aries_cloudagent.transport.outbound.message.OutboundMessage*)
 Add an outbound message to the queue.

Parameters

- **context** – The context of the request
- **outbound** – The outbound message to deliver

enqueue_webhook (*topic: str, payload: dict, endpoint: str, max_attempts: int = None*)
 Add a webhook to the queue.

Parameters

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

Raises *OutboundDeliveryError* – if the associated transport is not running

finished_deliver (*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage, completed: aries_cloudagent.utils.task_queue.CompletedTask*)
 Handle completion of queued message delivery.

finished_encode (*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage, completed: aries_cloudagent.utils.task_queue.CompletedTask*)
 Handle completion of queued message encoding.

flush()
 Wait for any queued messages to be delivered.

get_registered_transport_for_scheme (*scheme: str*) → *str*
 Find the registered transport ID for a given scheme.

get_running_transport_for_endpoint (*endpoint: str*)
 Find the running transport ID to use for a given endpoint.

get_running_transport_for_scheme (*scheme: str*) → *str*
 Find the running transport ID for a given scheme.

get_transport_instance (*transport_id: str*) → *aries_cloudagent.transport.outbound.base.BaseOutboundTransport*
 Get an instance of a running transport by ID.

perform_encode (*queued: aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage*)
 Perform message encoding.

process_queued() → *_asyncio.Task*
 Start the process to deliver queued messages if necessary.
 Returns: the current queue processing task or None

register (*module: str*) → *str*
 Register a new outbound transport by module path.

Parameters `module` – Module name to register

Raises

- `OutboundTransportRegistrationError` – If the imported class cannot be located
- `OutboundTransportRegistrationError` – If the imported class does not specify a `schemes` attribute
- `OutboundTransportRegistrationError` – If the scheme has already been registered

register_class (*transport_class: Type[aries_cloudagent.transport.outbound.base.BaseOutboundTransport],
transport_id: str = None*) → str

Register a new outbound transport class.

Parameters `transport_class` – Transport class to register

Raises

- `OutboundTransportRegistrationError` – If the imported class does not specify a `schemes` attribute
- `OutboundTransportRegistrationError` – If the scheme has already been registered

setup ()

Perform setup operations.

start ()

Start all transports and feed messages from the queue.

start_transport (*transport_id: str*)

Start a registered transport.

stop (*wait: bool = True*)

Stop all running transports.

class `aries_cloudagent.transport.outbound.manager.QueuedOutboundMessage` (*context: aries_cloudagent.config.injector.Injector, message: aries_cloudagent.transport.outbound.base.OutboundMessage, target_id: str, get: Callable[[str], aries_cloudagent.connection.Connection], transport_id: str*)

Bases: `object`

Class representing an outbound message pending delivery.

STATE_DELIVER = 'deliver'

STATE_DONE = 'done'

STATE_ENCODE = 'encode'

STATE_NEW = 'new'

STATE_PENDING = 'pending'

STATE_RETRY = 'retry'

aries_cloudagent.transport.outbound.ws module

Websockets outbound transport.

```
class aries_cloudagent.transport.outbound.ws.WsTransport
    Bases: aries_cloudagent.transport.outbound.base.BaseOutboundTransport
    Websockets outbound transport class.

    handle_message (payload: Union[str, bytes], endpoint: str)
        Handle message from queue.

        Parameters message – OutboundMessage to send over transport implementation

    schemes = ('ws', 'wss')

    start ()
        Start the outbound transport.

    stop ()
        Stop the outbound transport.
```

1.1.12 aries_cloudagent.verifier package

Submodules

aries_cloudagent.verifier.base module

Base Verifier class.

```
class aries_cloudagent.verifier.base.BaseVerifier
    Bases: abc.ABC
    Base class for verifier.
```

aries_cloudagent.verifier.indy module

Indy verifier implementation.

```
class aries_cloudagent.verifier.indy.IndyVerifier (wallet)
    Bases: aries_cloudagent.verifier.base.BaseVerifier
    Indy holder class.

    verify_presentation (presentation_request, presentation, schemas, credential_definitions) → bool
        Verify a presentation.

        Parameters
        • presentation_request – Presentation request data
        • presentation – Presentation data
        • schemas – Schema data
        • credential_definitions – credential definition data
```

1.1.13 aries_cloudagent.wallet package

Abstract and Indy wallet handling.

Submodules

aries_cloudagent.wallet.base module

Wallet base class.

class aries_cloudagent.wallet.base.**BaseWallet** (*config: dict*)

Bases: `abc.ABC`

Abstract wallet interface.

WALLET_TYPE = None

close ()

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

create_local_did (*seed: str = None, did: str = None, metadata: dict = None*) →
aries_cloudagent.wallet.base.DIDInfo

Create and store a new local DID.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns The created *DIDInfo*

create_public_did (*seed: str = None, did: str = None, metadata: dict = {}*) →
aries_cloudagent.wallet.base.DIDInfo

Create and store a new public DID.

Implicitly flags all other dids as not public.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns The created *DIDInfo*

create_signing_key (*seed: str = None, metadata: dict = None*) →
aries_cloudagent.wallet.base.KeyInfo

Create a new public/private signing keypair.

Parameters

- **seed** – Optional seed allowing deterministic key creation
- **metadata** – Optional metadata to store with the keypair

Returns A *KeyInfo* representing the new record

created

Check whether the wallet was created on the last open call.

get_local_did (*did: str*) → aries_cloudagent.wallet.base.DIDInfo
Find info for a local DID.

Parameters **did** – The DID to get info for

Returns A *DIDInfo* instance for the DID

get_local_did_for_verkey (*verkey: str*) → aries_cloudagent.wallet.base.DIDInfo
Resolve a local DID from a verkey.

Parameters **verkey** – Verkey to get DID info for

Returns A *DIDInfo* instance for the DID

get_local_dids () → Sequence[aries_cloudagent.wallet.base.DIDInfo]
Get list of defined local DIDs.

Returns A list of *DIDInfo* instances

get_public_did () → aries_cloudagent.wallet.base.DIDInfo
Retrieve the public did.

Returns The created *DIDInfo*

get_signing_key (*verkey: str*) → aries_cloudagent.wallet.base.KeyInfo
Fetch info for a signing keypair.

Parameters **verkey** – The verification key of the keypair

Returns A *KeyInfo* representing the keypair

handle
Get internal wallet reference.

Returns Defaults to None

name
Accessor for the wallet name.

Returns Defaults to None

open ()
Open wallet, removing and/or creating it if so configured.

opened
Check whether wallet is currently open.

Returns Defaults to False

pack_message (*message: str, to_verkeys: Sequence[str], from_verkey: str = None*) → bytes
Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – The verkeys to pack the message for
- **from_verkey** – The sender verkey

Returns The packed message

replace_local_did_metadata (*did: str, metadata: dict*)
Replace the metadata associated with a local DID.

Parameters

- **did** – DID to replace metadata for

- **metadata** – The new metadata

replace_signing_key_metadata (*verkey: str, metadata: dict*)

Replace the metadata associated with a signing keypair.

Parameters

- **verkey** – The verification key of the keypair
- **metadata** – The new metadata to store

set_public_did (*did: str*) → *aries_cloudagent.wallet.base.DIDInfo*

Assign the public did.

Returns The created *DIDInfo*

sign_message (*message: bytes, from_verkey: str*) → *bytes*

Sign a message using the private key associated with a given verkey.

Parameters

- **message** – The message to sign
- **from_verkey** – Sign using the private key related to this verkey

Returns The signature

type

Accessor for the wallet type.

Returns Defaults to None

unpack_message (*enc_message: bytes*) -> (*<class 'str'>, <class 'str'>, <class 'str'>*)

Unpack a message.

Parameters **enc_message** – The encrypted message

Returns (message, from_verkey, to_verkey)

Return type A tuple

verify_message (*message: bytes, signature: bytes, from_verkey: str*) → *bool*

Verify a signature against the public key of the signer.

Parameters

- **message** – The message to verify
- **signature** – The signature to verify
- **from_verkey** – Verkey to use in verification

Returns True if verified, else False

class *aries_cloudagent.wallet.base.DIDInfo* (*did, verkey, metadata*)

Bases: *tuple*

did

Alias for field number 0

metadata

Alias for field number 2

verkey

Alias for field number 1

class *aries_cloudagent.wallet.base.KeyInfo* (*verkey, metadata*)

Bases: *tuple*

metadata

Alias for field number 1

verkey

Alias for field number 0

aries_cloudagent.wallet.basic module

In-memory implementation of BaseWallet interface.

class aries_cloudagent.wallet.basic.**BasicWallet** (*config: dict = None*)Bases: *aries_cloudagent.wallet.base.BaseWallet*

In-memory wallet implementation.

WALLET_TYPE = 'basic'**close()**

Not applicable to in-memory wallet.

create_local_did (*seed: str = None, did: str = None, metadata: dict = None*) →*aries_cloudagent.wallet.base.DIDInfo*

Create and store a new local DID.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns A *DIDInfo* instance representing the created DID**Raises** *WalletDuplicateError* – If the DID already exists in the wallet**create_signing_key** (*seed: str = None, metadata: dict = None*) →*aries_cloudagent.wallet.base.KeyInfo*

Create a new public/private signing keypair.

Parameters

- **seed** – Seed to use for signing key
- **metadata** – Optional metadata to store with the keypair

Returns A *KeyInfo* representing the new record**Raises** *WalletDuplicateError* – If the resulting verkey already exists in the wallet**created**

Check whether the wallet was created on the last open call.

get_local_did (*did: str*) → *aries_cloudagent.wallet.base.DIDInfo*

Find info for a local DID.

Parameters **did** – The DID to get info for**Returns** A *DIDInfo* instance representing the found DID**Raises** *WalletNotFoundError* – If the DID is not found**get_local_did_for_verkey** (*verkey: str*) → *aries_cloudagent.wallet.base.DIDInfo*

Resolve a local DID from a verkey.

Parameters **verkey** – The verkey to get the local DID for

Returns A *DIDInfo* instance representing the found DID

Raises *WalletNotFoundError* – If the verkey is not found

get_local_dids () → Sequence[aries_cloudagent.wallet.base.DIDInfo]
Get list of defined local DIDs.

Returns A list of locally stored DIDs as *DIDInfo* instances

get_signing_key (verkey: str) → aries_cloudagent.wallet.base.KeyInfo
Fetch info for a signing keypair.

Parameters **verkey** – The verification key of the keypair

Returns A *KeyInfo* representing the keypair

Raises *WalletNotFoundError* – if no keypair is associated with the verification key

name
Accessor for the wallet name.

open ()
Not applicable to in-memory wallet.

opened
Check whether wallet is currently open.

Returns True

pack_message (message: str, to_verkeys: Sequence[str], from_verkey: str = None) → bytes
Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – List of verkeys to pack for
- **from_verkey** – Sender verkey to pack from

Returns The resulting packed message bytes

replace_local_did_metadata (did: str, metadata: dict)
Replace metadata for a local DID.

Parameters

- **did** – The DID to replace metadata for
- **metadata** – The new metadata

Raises *WalletNotFoundError* – If the DID doesn't exist

replace_signing_key_metadata (verkey: str, metadata: dict)
Replace the metadata associated with a signing keypair.

Parameters

- **verkey** – The verification key of the keypair
- **metadata** – The new metadata to store

Raises *WalletNotFoundError* – if no keypair is associated with the verification key

sign_message (message: bytes, from_verkey: str) → bytes
Sign a message using the private key associated with a given verkey.

Parameters

- **message** – Message bytes to sign
- **from_verkey** – The verkey to use to sign

Returns A signature

Raises

- `WalletError` – If the message is not provided
- `WalletError` – If the verkey is not provided

unpack_message (*enc_message: bytes*) -> (<class 'str'>, <class 'str'>, <class 'str'>)

Unpack a message.

Parameters **enc_message** – The packed message bytes

Returns (message, from_verkey, to_verkey)

Return type A tuple

Raises

- `WalletError` – If the message is not provided
- `WalletError` – If there is a problem unpacking the message

verify_message (*message: bytes, signature: bytes, from_verkey: str*) → bool

Verify a signature against the public key of the signer.

Parameters

- **message** – Message to verify
- **signature** – Signature to verify
- **from_verkey** – Verkey to use in verification

Returns True if verified, else False

Raises

- `WalletError` – If the verkey is not provided
- `WalletError` – If the signature is not provided
- `WalletError` – If the message is not provided

aries_cloudagent.wallet.crypto module

Cryptography functions used by BasicWallet.

```
class aries_cloudagent.wallet.crypto.PackMessageSchema (*,      only=None,      ex-
                                                         clude=(), many=False, con-
                                                         text=None,  load_only=(),
                                                         dump_only=(),      par-
                                                         tial=False,        un-
                                                         known=None)
```

Bases: `marshmallow.schema.Schema`

Packed message schema.

opts = <marshmallow.schema.SchemaOpts object>

```
class aries_cloudagent.wallet.crypto.PackRecipientHeaderSchema (*, only=None,
                                                                exclude=(),
                                                                many=False,
                                                                context=None,
                                                                load_only=(),
                                                                dump_only=(),
                                                                partial=False,
                                                                unknown=None)
```

Bases: `marshmallow.schema.Schema`

Packed recipient header schema.

opts = `<marshmallow.schema.SchemaOpts object>`

```
class aries_cloudagent.wallet.crypto.PackRecipientSchema (*, only=None,
                                                            exclude=(),
                                                            many=False,
                                                            context=None,
                                                            load_only=(),
                                                            dump_only=(),
                                                            partial=False,
                                                            unknown=None)
```

Bases: `marshmallow.schema.Schema`

Packed recipient schema.

opts = `<marshmallow.schema.SchemaOpts object>`

```
class aries_cloudagent.wallet.crypto.PackRecipientsSchema (*, only=None,
                                                            exclude=(),
                                                            many=False,
                                                            context=None,
                                                            load_only=(),
                                                            dump_only=(),
                                                            partial=False,
                                                            unknown=None)
```

Bases: `marshmallow.schema.Schema`

Packed recipients schema.

opts = `<marshmallow.schema.SchemaOpts object>`

```
aries_cloudagent.wallet.crypto.create_keypair (seed: bytes = None) → Tuple[bytes, bytes]
```

Create a public and private signing keypair from a seed value.

Parameters **seed** – Seed for keypair

Returns A tuple of (public key, secret key)

```
aries_cloudagent.wallet.crypto.decode_pack_message (enc_message: bytes, find_key: Callable) → Tuple[str, Optional[str], str]
```

Decode a packed message.

Disassemble and unencrypt a packed message, returning the message content, verification key of the sender (if available), and verification key of the recipient.

Parameters

- **enc_message** – The encrypted message
- **find_key** – Function to retrieve private key

Returns A tuple of (message, sender_vk, recip_vk)

Raises

- `ValueError` – If the packed message is invalid
- `ValueError` – If the packed message recipients are invalid
- `ValueError` – If the pack algorithm is unsupported
- `ValueError` – If the sender's public key was not provided

`aries_cloudagent.wallet.crypto.decode_pack_message_outer` (*enc_message: bytes*) →
 Tuple[dict, dict, bool]

Decode the outer wrapper of a packed message and extract the recipients.

Parameters `enc_message` – The encrypted message

Returns: a tuple of the decoded wrapper, recipients, and authcrypt flag

`aries_cloudagent.wallet.crypto.decode_pack_message_payload` (*wrapper: dict, payload_key: bytes*) →
 str

Decode the payload of a packed message once the CEK is known.

Parameters

- **wrapper** – The decoded message wrapper
- **payload_key** – The decrypted payload key

`aries_cloudagent.wallet.crypto.decrypt_plaintext` (*ciphertext: bytes, recips_bin: bytes, nonce: bytes, key: bytes*) → str

Decrypt the payload of a packed message.

Parameters

- **ciphertext** –
- **recips_bin** –
- **nonce** –
- **key** –

Returns The decrypted string

`aries_cloudagent.wallet.crypto.encode_pack_message` (*message: str, to_verkeys: Sequence[bytes], from_secret: bytes = None*) → bytes

Assemble a packed message for a set of recipients, optionally including the sender.

Parameters

- **message** – The message to pack
- **to_verkeys** – The verkeys to pack the message for
- **from_secret** – The sender secret

Returns The encoded message

`aries_cloudagent.wallet.crypto.encrypt_plaintext` (*message: str, add_data: bytes, key: bytes*) → Tuple[bytes, bytes, bytes]

Encrypt the payload of a packed message.

Parameters

- **message** – Message to encrypt
- **add_data** –

- **key** – Key used for encryption

Returns A tuple of (ciphertext, nonce, tag)

`aries_cloudagent.wallet.crypto.extract_pack_recipients` (*recipients: Sequence[dict]*)
→ dict

Extract the pack message recipients into a dict indexed by verkey.

Parameters **recipients** – Recipients to locate

Raises `ValueError` – If the recipients block is mal-formatted

`aries_cloudagent.wallet.crypto.extract_payload_key` (*sender_cek: dict, recip_secret: bytes*) → Tuple[bytes, str]

Extract the payload key from pack recipient details.

Returns: A tuple of the CEK and sender verkey

`aries_cloudagent.wallet.crypto.prepare_pack_recipient_keys` (*to_verkeys: Sequence[bytes], from_secret: bytes = None*) → Tuple[str, bytes]

Assemble the recipients block of a packed message.

Parameters

- **to_verkeys** – Verkeys of recipients
- **from_secret** – Secret to use for signing keys

Returns A tuple of (json result, key)

`aries_cloudagent.wallet.crypto.random_seed`() → bytes
Generate a random seed value.

Returns A new random seed

`aries_cloudagent.wallet.crypto.seed_to_did` (*seed: str*) → str
Derive a DID from a seed value.

Parameters **seed** – The seed to derive

Returns The DID derived from the seed

`aries_cloudagent.wallet.crypto.sign_message` (*message: bytes, secret: bytes*) → bytes
Sign a message using a private signing key.

Parameters

- **message** – The message to sign
- **secret** – The private signing key

Returns The signature

`aries_cloudagent.wallet.crypto.sign_pk_from_sk` (*secret: bytes*) → bytes
Extract the verkey from a secret signing key.

`aries_cloudagent.wallet.crypto.validate_seed` (*seed: (<class 'str'>, <class 'bytes'>)*) → bytes
Convert a seed parameter to standard format and check length.

Parameters **seed** – The seed to validate

Returns The validated and encoded seed

`aries_cloudagent.wallet.crypto.verify_signed_message` (*signed: bytes, verkey: bytes*)
→ bool

Verify a signed message according to a public verification key.

Parameters

- **signed** – The signed message
- **verkey** – The verkey to use in verification

Returns True if verified, else False

aries_cloudagent.wallet.error module

Wallet-related exceptions.

exception `aries_cloudagent.wallet.error.WalletDuplicateError` (**args, error_code: str = None, **kwargs*)

Bases: `aries_cloudagent.wallet.error.WalletError`

Duplicate record exception.

exception `aries_cloudagent.wallet.error.WalletError` (**args, error_code: str = None, **kwargs*)

Bases: `aries_cloudagent.core.error.BaseError`

General wallet exception.

exception `aries_cloudagent.wallet.error.WalletNotFoundError` (**args, error_code: str = None, **kwargs*)

Bases: `aries_cloudagent.wallet.error.WalletError`

Record not found exception.

aries_cloudagent.wallet.indy module

Indy implementation of BaseWallet interface.

class `aries_cloudagent.wallet.indy.IndyWallet` (*config: dict = None*)
Bases: `aries_cloudagent.wallet.base.BaseWallet`

Indy wallet implementation.

DEFAULT_FRESHNESS = 0

DEFAULT_KEY = ''

DEFAULT_KEY_DERIVATION = 'ARGON2I_MOD'

DEFAULT_NAME = 'default'

DEFAULT_STORAGE_TYPE = None

KEY_DERIVATION_ARGON2I_INT = 'ARGON2I_INT'

KEY_DERIVATION_ARGON2I_MOD = 'ARGON2I_MOD'

KEY_DERIVATION_RAW = 'RAW'

WALLET_TYPE = 'indy'

close()

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

create (*replace: bool = False*)

Create a new wallet.

Parameters **replace** – Removes the old wallet if True

Raises

- `WalletError` – If there was a problem removing the wallet
- `WalletError` – IF there was a libindy error

create_local_did (*seed: str = None, did: str = None, metadata: dict = None*) → `aries_cloudagent.wallet.base.DIDInfo`

Create and store a new local DID.

Parameters

- **seed** – Optional seed to use for did
- **did** – The DID to use
- **metadata** – Metadata to store with DID

Returns A *DIDInfo* instance representing the created DID

Raises

- `WalletDuplicateError` – If the DID already exists in the wallet
- `WalletError` – If there is a libindy error

create_signing_key (*seed: str = None, metadata: dict = None*) → `aries_cloudagent.wallet.base.KeyInfo`

Create a new public/private signing keypair.

Parameters

- **seed** – Seed for key
- **metadata** – Optional metadata to store with the keypair

Returns A *KeyInfo* representing the new record

Raises

- `WalletDuplicateError` – If the resulting verkey already exists in the wallet
- `WalletError` – If there is a libindy error

created

Check whether the wallet was created on the last open call.

classmethod generate_wallet_key (*seed: str = None*) → str

Generate a raw Indy wallet key.

get_credential_definition_tag_policy (*credential_definition_id: str*)

Return the tag policy for a given credential definition ID.

get_local_did (*did: str*) → `aries_cloudagent.wallet.base.DIDInfo`

Find info for a local DID.

Parameters **did** – The DID to get info for

Returns A *DIDInfo* instance representing the found DID

Raises

- `WalletNotFoundError` – If the DID is not found
- `WalletError` – If there is a libindy error

get_local_did_for_verkey (*verkey: str*) → `aries_cloudagent.wallet.base.DIDInfo`
Resolve a local DID from a verkey.

Parameters **verkey** – The verkey to get the local DID for

Returns A *DIDInfo* instance representing the found DID

Raises `WalletNotFoundError` – If the verkey is not found

get_local_dids () → `Sequence[aries_cloudagent.wallet.base.DIDInfo]`
Get list of defined local DIDs.

Returns A list of locally stored DIDs as *DIDInfo* instances

get_signing_key (*verkey: str*) → `aries_cloudagent.wallet.base.KeyInfo`
Fetch info for a signing keypair.

Parameters **verkey** – The verification key of the keypair

Returns A *KeyInfo* representing the keypair

Raises

- `WalletNotFoundError` – If no keypair is associated with the verification key
- `WalletError` – If there is a libindy error

handle

Get internal wallet reference.

Returns A handle to the wallet

master_secret_id

Accessor for the master secret id.

Returns The master secret id

name

Accessor for the wallet name.

Returns The wallet name

open ()

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

Raises

- `WalletError` – If wallet not found after creation
- `WalletNotFoundError` – If the wallet is not found
- `WalletError` – If the wallet is already open
- `WalletError` – If there is a libindy error

opened

Check whether wallet is currently open.

Returns True if open, else False

pack_message (*message: str, to_verkeys: Sequence[str], from_verkey: str = None*) → bytes
Pack a message for one or more recipients.

Parameters

- **message** – The message to pack
- **to_verkeys** – List of verkeys to pack for
- **from_verkey** – Sender verkey to pack from

Returns The resulting packed message bytes

Raises

- `WalletError` – If no message is provided
- `WalletError` – If a libindy error occurs

remove()

Remove an existing wallet.

Raises

- `WalletNotFoundError` – If the wallet could not be found
- `WalletError` – If there was an libindy error

replace_local_did_metadata (*did: str, metadata: dict*)

Replace metadata for a local DID.

Parameters

- **did** – The DID to replace metadata for
- **metadata** – The new metadata

replace_signing_key_metadata (*verkey: str, metadata: dict*)

Replace the metadata associated with a signing keypair.

Parameters

- **verkey** – The verification key of the keypair
- **metadata** – The new metadata to store

Raises `WalletNotFoundError` – if no keypair is associated with the verification key

set_credential_definition_tag_policy (*credential_definition_id: str, taggables: Sequence[str] = None, retroactive: bool = True*)

Set the tag policy for a given credential definition ID.

Parameters

- **credential_definition_id** – The ID of the credential definition
- **taggables** – A sequence of string values representing attribute names
- **retroactive** – Whether to apply the policy to previously-stored credentials

sign_message (*message: bytes, from_verkey: str*) → bytes

Sign a message using the private key associated with a given verkey.

Parameters

- **message** – Message bytes to sign
- **from_verkey** – The verkey to use to sign

Returns A signature

Raises

- `WalletError` – If the message is not provided

- `WalletError` – If the verkey is not provided
- `WalletError` – If a libindy error occurs

unpack_message (*enc_message: bytes*) -> (<class 'str'>, <class 'str'>, <class 'str'>)

Unpack a message.

Parameters *enc_message* – The packed message bytes

Returns (message, from_verkey, to_verkey)

Return type A tuple

Raises

- `WalletError` – If the message is not provided
- `WalletError` – If a libindy error occurs

verify_message (*message: bytes, signature: bytes, from_verkey: str*) → bool

Verify a signature against the public key of the signer.

Parameters

- **message** – Message to verify
- **signature** – Signature to verify
- **from_verkey** – Verkey to use in verification

Returns True if verified, else False

Raises

- `WalletError` – If the verkey is not provided
- `WalletError` – If the signature is not provided
- `WalletError` – If the message is not provided
- `WalletError` – If a libindy error occurs

aries_cloudagent.wallet.provider module

Default wallet provider classes.

class `aries_cloudagent.wallet.provider.WalletProvider`

Bases: `aries_cloudagent.config.base.BaseProvider`

Provider for the default configurable wallet classes.

WALLET_TYPES = {'basic': 'aries_cloudagent.wallet.basic.BasicWallet', 'indy': 'aries

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

Create and open the wallet instance.

aries_cloudagent.wallet.routes module

Wallet admin routes.

```
class aries_cloudagent.wallet.routes.DIDListSchema (*, only=None, exclude=(),
many=False, context=None,
load_only=(), dump_only=(),
partial=False, unknown=None)
```

Bases: `marshmallow.schema.Schema`

Result schema for connection list.

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
class aries_cloudagent.wallet.routes.DIDResultSchema (*, only=None, ex-
clude=(), many=False, con-
text=None, load_only=(),
dump_only=(), partial=False,
unknown=None)
```

Bases: `marshmallow.schema.Schema`

Result schema for a DID.

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
class aries_cloudagent.wallet.routes.DIDSchema (*, only=None, exclude=(), many=False,
context=None, load_only=(),
dump_only=(), partial=False, un-
known=None)
```

Bases: `marshmallow.schema.Schema`

Result schema for a DID.

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
class aries_cloudagent.wallet.routes.GetTagPolicyResultSchema (*, only=None,
exclude=(),
many=False,
context=None,
load_only=(),
dump_only=(),
partial=False,
unknown=None)
```

Bases: `marshmallow.schema.Schema`

Result schema for tagging policy get request.

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
class aries_cloudagent.wallet.routes.SetTagPolicyRequestSchema (*, only=None,
exclude=(),
many=False,
context=None,
load_only=(),
dump_only=(),
par-
tial=False, un-
known=None)
```

Bases: `marshmallow.schema.Schema`

Request schema for tagging policy set request.

```
opts = <marshmallow.schema.SchemaOpts object>
```

```
aries_cloudagent.wallet.routes.format_did_info (info: aries_cloudagent.wallet.base.DIDInfo)
Serialize a DIDInfo object.
```

`aries_cloudagent.wallet.routes.register` (*app: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b66c978>*)

Register routes.

`aries_cloudagent.wallet.routes.wallet_create_did` (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b66c978>*)

Request handler for creating a new wallet DID.

Parameters `request` – aiohttp request object

Returns The DID info

`aries_cloudagent.wallet.routes.wallet_did_list` (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b66c978>*)

Request handler for searching wallet DIDs.

Parameters `request` – aiohttp request object

Returns The DID list response

`aries_cloudagent.wallet.routes.wallet_get_public_did` (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b66c978>*)

Request handler for fetching the current public DID.

Parameters `request` – aiohttp request object

Returns The DID info

`aries_cloudagent.wallet.routes.wallet_get_tagging_policy` (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b66c978>*)

Request handler for getting the tag policy associated with a cred def.

Parameters `request` – aiohttp request object

Returns A JSON object containing the tagging policy

`aries_cloudagent.wallet.routes.wallet_set_public_did` (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b66c978>*)

Request handler for setting the current public DID.

Parameters `request` – aiohttp request object

Returns The updated DID info

`aries_cloudagent.wallet.routes.wallet_set_tagging_policy` (*request: <sphinx.ext.autodoc.importer._MockObject object at 0x7f538b66c978>*)

Request handler for setting the tag policy associated with a cred def.

Parameters `request` – aiohttp request object

Returns An empty JSON response

aries_cloudagent.wallet.util module

Wallet utility functions.

`aries_cloudagent.wallet.util.b58_to_bytes(val: str) → bytes`

Convert a base 58 string to bytes.

`aries_cloudagent.wallet.util.b64_to_bytes(val: str, urlsafe=False) → bytes`

Convert a base 64 string to bytes.

`aries_cloudagent.wallet.util.b64_to_str(val: str, urlsafe=False, encoding=None) → str`

Convert a base 64 string to string on input encoding (default utf-8).

`aries_cloudagent.wallet.util.bytes_to_b58(val: bytes) → str`

Convert a byte string to base 58.

`aries_cloudagent.wallet.util.bytes_to_b64(val: bytes, urlsafe=False, pad=True) → str`

Convert a byte string to base 64.

`aries_cloudagent.wallet.util.pad(val: str) → str`

Pad base64 values if need be: JWT calls to omit trailing padding.

`aries_cloudagent.wallet.util.set_urlsafe_b64(val: str, urlsafe: bool = True) → str`

Set URL safety in base64 encoding.

`aries_cloudagent.wallet.util.str_to_b64(val: str, urlsafe=False, encoding=None, pad=True) → str`

Convert a string to base64 string on input encoding (default utf-8).

`aries_cloudagent.wallet.util.unpad(val: str) → str`

Remove padding from base64 values if need be.

1.2 Submodules

1.3 `aries_cloudagent.classloader` module

1.4 `aries_cloudagent.conductor` module

1.5 `aries_cloudagent.defaults` module

1.6 `aries_cloudagent.dispatcher` module

1.7 `aries_cloudagent.error` module

1.8 `aries_cloudagent.postgres` module

1.9 `aries_cloudagent.stats` module

1.10 `aries_cloudagent.task_processor` module

1.11 `aries_cloudagent.version` module

Library version information.

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

Python Module Index

a

`aries_cloudagent`, 3
`aries_cloudagent.admin`, 3
`aries_cloudagent.admin.base_server`, 3
`aries_cloudagent.admin.error`, 4
`aries_cloudagent.admin.server`, 4
`aries_cloudagent.cache`, 6
`aries_cloudagent.cache.base`, 6
`aries_cloudagent.cache.basic`, 7
`aries_cloudagent.commands`, 7
`aries_cloudagent.commands.help`, 8
`aries_cloudagent.commands.provision`, 8
`aries_cloudagent.commands.start`, 8
`aries_cloudagent.config`, 9
`aries_cloudagent.config.argparse`, 9
`aries_cloudagent.config.base`, 11
`aries_cloudagent.config.base_context`, 12
`aries_cloudagent.config.default_context`, 13
`aries_cloudagent.config.error`, 13
`aries_cloudagent.config.injection_context`, 13
`aries_cloudagent.config.injector`, 15
`aries_cloudagent.config.ledger`, 15
`aries_cloudagent.config.logging`, 15
`aries_cloudagent.config.provider`, 16
`aries_cloudagent.config.settings`, 17
`aries_cloudagent.config.util`, 18
`aries_cloudagent.config.wallet`, 18
`aries_cloudagent.holder`, 18
`aries_cloudagent.holder.base`, 18
`aries_cloudagent.holder.indy`, 18
`aries_cloudagent.issuer`, 20
`aries_cloudagent.issuer.base`, 20
`aries_cloudagent.issuer.indy`, 20
`aries_cloudagent.issuer.util`, 21
`aries_cloudagent.ledger`, 21
`aries_cloudagent.ledger.base`, 21
`aries_cloudagent.ledger.error`, 22
`aries_cloudagent.ledger.indy`, 23
`aries_cloudagent.ledger.provider`, 25
`aries_cloudagent.messaging`, 25
`aries_cloudagent.messaging.agent_message`, 52
`aries_cloudagent.messaging.base_handler`, 54
`aries_cloudagent.messaging.credential_definitions`, 28
`aries_cloudagent.messaging.credential_definitions...`, 28
`aries_cloudagent.messaging.decorators`, 32
`aries_cloudagent.messaging.decorators.base`, 32
`aries_cloudagent.messaging.decorators.default`, 33
`aries_cloudagent.messaging.decorators.localization`, 33
`aries_cloudagent.messaging.decorators.signature_de`, 34
`aries_cloudagent.messaging.decorators.thread_decor`, 36
`aries_cloudagent.messaging.decorators.timing_decor`, 37
`aries_cloudagent.messaging.decorators.transport_de`, 39
`aries_cloudagent.messaging.error`, 55
`aries_cloudagent.messaging.models`, 41
`aries_cloudagent.messaging.models.base`, 41
`aries_cloudagent.messaging.models.base_record`, 43
`aries_cloudagent.messaging.request_context`, 55
`aries_cloudagent.messaging.responder`, 56
`aries_cloudagent.messaging.schemas`, 49
`aries_cloudagent.messaging.schemas.routes`,

49

- `aries_cloudagent.messaging.util`, 58
- `aries_cloudagent.storage`, 58
 - `aries_cloudagent.storage.base`, 58
 - `aries_cloudagent.storage.basic`, 61
 - `aries_cloudagent.storage.error`, 63
 - `aries_cloudagent.storage.indy`, 63
 - `aries_cloudagent.storage.provider`, 66
 - `aries_cloudagent.storage.record`, 66
- `aries_cloudagent.transport`, 66
 - `aries_cloudagent.transport.inbound`, 66
 - `aries_cloudagent.transport.inbound.base`, 66
 - `aries_cloudagent.transport.inbound.http`, 68
 - `aries_cloudagent.transport.inbound.manager`, 69
 - `aries_cloudagent.transport.inbound.ws`, 70
 - `aries_cloudagent.transport.outbound`, 71
 - `aries_cloudagent.transport.outbound.base`, 71
 - `aries_cloudagent.transport.outbound.http`, 72
 - `aries_cloudagent.transport.outbound.manager`, 72
 - `aries_cloudagent.transport.outbound.ws`, 75
- `aries_cloudagent.verifier`, 75
 - `aries_cloudagent.verifier.base`, 75
 - `aries_cloudagent.verifier.indy`, 75
- `aries_cloudagent.version`, 92
- `aries_cloudagent.wallet`, 76
 - `aries_cloudagent.wallet.base`, 76
 - `aries_cloudagent.wallet.basic`, 79
 - `aries_cloudagent.wallet.crypto`, 81
 - `aries_cloudagent.wallet.error`, 85
 - `aries_cloudagent.wallet.indy`, 85
 - `aries_cloudagent.wallet.provider`, 89
 - `aries_cloudagent.wallet.routes`, 89
 - `aries_cloudagent.wallet.util`, 91

A

`accept_taa()` (in module `aries_cloudagent.config.ledger`), 15
`accept_txn_author_agreement()` (`aries_cloudagent.ledger.base.BaseLedger` method), 21
`accept_txn_author_agreement()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 23
`acquire()` (`aries_cloudagent.cache.base.BaseCache` method), 6
`add_arguments()` (`aries_cloudagent.config.argparse.AdminGroup` method), 9
`add_arguments()` (`aries_cloudagent.config.argparse.ArgumentGroup` method), 9
`add_arguments()` (`aries_cloudagent.config.argparse.DebugGroup` method), 9
`add_arguments()` (`aries_cloudagent.config.argparse.GeneralGroup` method), 9
`add_arguments()` (`aries_cloudagent.config.argparse.LedgerGroup` method), 10
`add_arguments()` (`aries_cloudagent.config.argparse.LoggingGroup` method), 10
`add_arguments()` (`aries_cloudagent.config.argparse.ProtocolGroup` method), 10
`add_arguments()` (`aries_cloudagent.config.argparse.TransportGroup` method), 10
`add_arguments()` (`aries_cloudagent.config.argparse.WalletGroup` method), 11
`add_model()` (`aries_cloudagent.messaging.decorators.base.BaseDecoratorSet` method), 32
`add_record()` (`aries_cloudagent.storage.base.BaseStorage` method), 59
`add_record()` (`aries_cloudagent.storage.basic.BasicStorage` method), 61
`add_record()` (`aries_cloudagent.storage.indy.IndyStorage` method), 63
`add_webhook_target()` (`aries_cloudagent.admin.base_server.BaseAdminServer` method), 3
`add_webhook_target()` (`aries_cloudagent.admin.server.AdminServer` method), 5
`AdminError`, 4
`AdminGroup` (class in `aries_cloudagent.config.argparse`), 9
`AdminModulesSchema` (class in `aries_cloudagent.admin.server`), 4
`AdminResponder` (class in `aries_cloudagent.admin.server`), 4
`AdminServer` (class in `aries_cloudagent.admin.server`), 4
`AdminSetupError`, 4
`AdminStatusSchema` (class in `aries_cloudagent.admin.server`), 5
`AgentMessage` (class in `aries_cloudagent.messaging.agent_message`), 52
`AgentMessage.Meta` (class in `aries_cloudagent.messaging.agent_message`), 52
`AgentMessageError`, 53
`AgentMessageSchema` (class in `aries_cloudagent.messaging.agent_message`), 53
`AgentMessageSchema.Meta` (class in `aries_cloudagent.messaging.agent_message`), 53
`ArgParseError`, 13
`ArgumentGroup` (class in `aries_cloudagent.config.argparse`), 9
`aries_cloudagent` (module), 3
`aries_cloudagent.admin` (module), 3
`aries_cloudagent.admin.base_server` (module), 3
`aries_cloudagent.admin.error` (module), 4
`aries_cloudagent.admin.server` (module), 4
`aries_cloudagent.cache` (module), 6
`aries_cloudagent.cache.base` (module), 6

[aries_cloudagent.cache.basic \(module\), 7](#)
[aries_cloudagent.commands \(module\), 7](#)
[aries_cloudagent.commands.help \(module\), 8](#)
[aries_cloudagent.commands.provision \(module\), 8](#)
[aries_cloudagent.commands.start \(module\), 8](#)
[aries_cloudagent.config \(module\), 9](#)
[aries_cloudagent.config.argparse \(module\), 9](#)
[aries_cloudagent.config.base \(module\), 11](#)
[aries_cloudagent.config.base_context \(module\), 12](#)
[aries_cloudagent.config.default_context \(module\), 13](#)
[aries_cloudagent.config.error \(module\), 13](#)
[aries_cloudagent.config.injection_context \(module\), 13](#)
[aries_cloudagent.config.injector \(module\), 15](#)
[aries_cloudagent.config.ledger \(module\), 15](#)
[aries_cloudagent.config.logging \(module\), 15](#)
[aries_cloudagent.config.provider \(module\), 16](#)
[aries_cloudagent.config.settings \(module\), 17](#)
[aries_cloudagent.config.util \(module\), 18](#)
[aries_cloudagent.config.wallet \(module\), 18](#)
[aries_cloudagent.holder \(module\), 18](#)
[aries_cloudagent.holder.base \(module\), 18](#)
[aries_cloudagent.holder.indy \(module\), 18](#)
[aries_cloudagent.issuer \(module\), 20](#)
[aries_cloudagent.issuer.base \(module\), 20](#)
[aries_cloudagent.issuer.indy \(module\), 20](#)
[aries_cloudagent.issuer.util \(module\), 21](#)
[aries_cloudagent.ledger \(module\), 21](#)
[aries_cloudagent.ledger.base \(module\), 21](#)
[aries_cloudagent.ledger.error \(module\), 22](#)
[aries_cloudagent.ledger.indy \(module\), 23](#)
[aries_cloudagent.ledger.provider \(module\), 25](#)
[aries_cloudagent.messaging \(module\), 25](#)
[aries_cloudagent.messaging.agent_message \(module\), 52](#)
[aries_cloudagent.messaging.base_handler \(module\), 54](#)
[aries_cloudagent.messaging.credential_definition \(module\), 28](#)
[aries_cloudagent.messaging.credential_definition_schemas \(module\), 28](#)
[aries_cloudagent.messaging.decorators \(module\), 32](#)
[aries_cloudagent.messaging.decorators.base \(module\), 32](#)
[aries_cloudagent.messaging.decorators.default \(module\), 33](#)
[aries_cloudagent.messaging.decorators.localization \(module\), 33](#)
[aries_cloudagent.messaging.decorators.signature_definition \(module\), 34](#)
[aries_cloudagent.messaging.decorators.thread_decorator \(module\), 36](#)
[aries_cloudagent.messaging.decorators.timing_decorator \(module\), 37](#)
[aries_cloudagent.messaging.decorators.transport_definition \(module\), 39](#)
[aries_cloudagent.messaging.error \(module\), 55](#)
[aries_cloudagent.messaging.models \(module\), 41](#)
[aries_cloudagent.messaging.models.base \(module\), 41](#)
[aries_cloudagent.messaging.models.base_record \(module\), 43](#)
[aries_cloudagent.messaging.request_context \(module\), 55](#)
[aries_cloudagent.messaging.responder \(module\), 56](#)
[aries_cloudagent.messaging.schemas \(module\), 49](#)
[aries_cloudagent.messaging.schemas.routes \(module\), 49](#)
[aries_cloudagent.messaging.util \(module\), 58](#)
[aries_cloudagent.storage \(module\), 58](#)
[aries_cloudagent.storage.base \(module\), 58](#)
[aries_cloudagent.storage.basic \(module\), 61](#)
[aries_cloudagent.storage.error \(module\), 63](#)
[aries_cloudagent.storage.indy \(module\), 63](#)
[aries_cloudagent.storage.provider \(module\), 66](#)
[aries_cloudagent.storage.record \(module\), 66](#)
[aries_cloudagent.transport \(module\), 66](#)
[aries_cloudagent.transport.inbound \(module\), 66](#)
[aries_cloudagent.transport.inbound.base \(module\), 66](#)
[aries_cloudagent.transport.inbound.http \(module\), 68](#)
[aries_cloudagent.transport.inbound.manager \(module\), 69](#)
[aries_cloudagent.transport.inbound.ws \(module\), 69](#)

(module), 70	BaseHolder (class in aries_cloudagent.holder.base), 18
aries_cloudagent.transport.outbound (module), 71	BaseInboundTransport (class in aries_cloudagent.transport.inbound.base), 66
aries_cloudagent.transport.outbound.base (module), 71	BaseInjector (class in aries_cloudagent.config.base), 11
aries_cloudagent.transport.outbound.httpBaseInjector (module), 72	BaseIssuer (class in aries_cloudagent.issuer.base), 20
aries_cloudagent.transport.outbound.manager (module), 72	BaseLedger (class in aries_cloudagent.ledger.base), 21
aries_cloudagent.transport.outbound.ws (module), 75	BaseModel (class in aries_cloudagent.messaging.models.base), 41
aries_cloudagent.verifier (module), 75	BaseModel.Meta (class in aries_cloudagent.messaging.models.base), 41
aries_cloudagent.verifier.base (module), 75	BaseModelError, 42
aries_cloudagent.verifier.indy (module), 75	BaseModelSchema (class in aries_cloudagent.messaging.models.base), 42
aries_cloudagent.version (module), 92	BaseModelSchema.Meta (class in aries_cloudagent.messaging.models.base), 42
aries_cloudagent.wallet (module), 76	BaseOutboundTransport (class in aries_cloudagent.transport.outbound.base), 71
aries_cloudagent.wallet.base (module), 76	BaseProvider (class in aries_cloudagent.config.base), 11
aries_cloudagent.wallet.basic (module), 79	BaseRecord (class in aries_cloudagent.messaging.models.base_record), 43
aries_cloudagent.wallet.crypto (module), 81	BaseRecord.Meta (class in aries_cloudagent.messaging.models.base_record), 46
aries_cloudagent.wallet.error (module), 85	BaseRecordSchema (class in aries_cloudagent.messaging.models.base_record), 46
aries_cloudagent.wallet.indy (module), 85	BaseRecordSchema.Meta (class in aries_cloudagent.messaging.models.base_record), 46
aries_cloudagent.wallet.provider (module), 89	BaseResponder (class in aries_cloudagent.messaging.responder), 56
aries_cloudagent.wallet.routes (module), 89	BaseSettings (class in aries_cloudagent.config.base), 11
aries_cloudagent.wallet.util (module), 91	BaseStorage (class in aries_cloudagent.storage.base), 58
assign_thread_from() (aries_cloudagent.messaging.agent_message.AgentMessage method), 52	BaseStorageRecordSearch (class in aries_cloudagent.storage.base), 60
assign_thread_id() (aries_cloudagent.messaging.agent_message.AgentMessage method), 52	BaseVerifier (class in aries_cloudagent.verifier.base), 75
available_commands() (in aries_cloudagent.commands), 7	BaseWallet (class in aries_cloudagent.wallet.base), 76
B	basic_tag_query_match() (in module)
b58_to_bytes() (in aries_cloudagent.wallet.util), 91	
b64_to_bytes() (in aries_cloudagent.wallet.util), 92	
b64_to_str() (in aries_cloudagent.wallet.util), 92	
BadLedgerRequestError, 22	
BaseAdminServer (class in aries_cloudagent.admin.base_server), 3	
BaseCache (class in aries_cloudagent.cache.base), 6	
BaseDecoratorSet (class in aries_cloudagent.messaging.decorators.base), 32	
BaseHandler (class in aries_cloudagent.messaging.base_handler), 54	

[aries_cloudagent.storage.basic](#), 63
[basic_tag_value_match\(\)](#) (in module [aries_cloudagent.storage.basic](#)), 63
[BasicCache](#) (class in [aries_cloudagent.cache.basic](#)), 7
[BasicStorage](#) (class in [aries_cloudagent.storage.basic](#)), 61
[BasicStorageRecordSearch](#) (class in [aries_cloudagent.storage.basic](#)), 62
[BasicWallet](#) (class in [aries_cloudagent.wallet.basic](#)), 79
[bind_instance\(\)](#) ([aries_cloudagent.config.injector.Injector](#) class method), 15
[bind_provider\(\)](#) ([aries_cloudagent.config.injector.Injector](#) class method), 15
[bind_providers\(\)](#) ([aries_cloudagent.config.default_context.DefaultContextBuilder](#) class method), 13
[build\(\)](#) ([aries_cloudagent.config.base_context.ContextBuilder](#) class method), 13
[build\(\)](#) ([aries_cloudagent.config.default_context.DefaultContextBuilder](#) class method), 13
[bytes_to_b58\(\)](#) (in module [aries_cloudagent.wallet.util](#)), 92
[bytes_to_b64\(\)](#) (in module [aries_cloudagent.wallet.util](#)), 92
[ByteSize](#) (class in [aries_cloudagent.config.util](#)), 18
C
[CACHE_ENABLED](#) ([aries_cloudagent.messaging.models.base_record.BaseRecord](#) class attribute), 43
[cache_key\(\)](#) ([aries_cloudagent.messaging.models.base_record.BaseRecord](#) class method), 43
[CACHE_TTL](#) ([aries_cloudagent.messaging.models.base_record.BaseRecord](#) class attribute), 43
[CachedProvider](#) (class in [aries_cloudagent.config.provider](#)), 16
[CacheError](#), 6
[CacheKeyLock](#) (class in [aries_cloudagent.cache.base](#)), 6
[canon\(\)](#) (in module [aries_cloudagent.messaging.util](#)), 58
[catalogs](#) ([aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecorator](#) class attribute), 34
[CATEGORIES](#) ([aries_cloudagent.config.argparse.AdminGroup](#) class attribute), 9
[CATEGORIES](#) ([aries_cloudagent.config.argparse.DebugGroup](#) class attribute), 9
[CATEGORIES](#) ([aries_cloudagent.config.argparse.GeneralGroup](#) class attribute), 9
[CATEGORIES](#) ([aries_cloudagent.config.argparse.LedgerGroup](#) class attribute), 10
[CATEGORIES](#) ([aries_cloudagent.config.argparse.LoggingGroup](#) class attribute), 10
[CATEGORIES](#) ([aries_cloudagent.config.argparse.ProtocolGroup](#) class attribute), 10
[CATEGORIES](#) ([aries_cloudagent.config.argparse.TransportGroup](#) class attribute), 10
[CATEGORIES](#) ([aries_cloudagent.config.argparse.WalletGroup](#) class attribute), 10
[check_dump_decorators\(\)](#) ([aries_cloudagent.messaging.agent_message.AgentMessageSchema](#) class method), 54
[check_existing_schema\(\)](#) ([aries_cloudagent.ledger.indy.IndyLedger](#) class method), 23
[check_pool_config\(\)](#) ([aries_cloudagent.ledger.indy.IndyLedger](#) class method), 23
[ClassProvider](#) (class in [aries_cloudagent.config.provider](#)), 16
[ClassProvider.Inject](#) (class in [aries_cloudagent.config.provider](#)), 16
[clear\(\)](#) ([aries_cloudagent.cache.base.BaseCache](#) class method), 6
[clear\(\)](#) ([aries_cloudagent.cache.basic.BasicCache](#) class method), 7
[clear_binding\(\)](#) ([aries_cloudagent.config.injector.Injector](#) class method), 15
[clear_cached\(\)](#) ([aries_cloudagent.messaging.models.base_record.BaseRecord](#) class method), 44
[clear_cached_key\(\)](#) ([aries_cloudagent.messaging.models.base_record.BaseRecord](#) class method), 44
[clear_value\(\)](#) ([aries_cloudagent.config.settings.Settings](#) class method), 17
[close\(\)](#) ([aries_cloudagent.ledger.indy.IndyLedger](#) class method), 23
[close\(\)](#) ([aries_cloudagent.storage.base.BaseStorageRecordSearch](#) class method), 60
[close\(\)](#) ([aries_cloudagent.storage.basic.BasicStorageRecordSearch](#) class method), 62
[close\(\)](#) ([aries_cloudagent.storage.indy.IndyStorageRecordSearch](#) class method), 65
[close\(\)](#) ([aries_cloudagent.wallet.base.BaseWallet](#) class method), 76
[close\(\)](#) ([aries_cloudagent.wallet.basic.BasicWallet](#) class method), 79
[close\(\)](#) ([aries_cloudagent.wallet.indy.IndyWallet](#) class method), 85
[closed_session\(\)](#) ([aries_cloudagent.transport.inbound.manager.InboundManager](#) class method), 69
[ClosedPoolError](#), 22
[collector](#) ([aries_cloudagent.transport.outbound.base.BaseOutboundTransport](#) class attribute), 71
[common_config\(\)](#) (in module [aries_cloudagent.config.util](#)), 18
[ConfigError](#), 12
[configure\(\)](#) ([aries_cloudagent.config.logging.LoggingConfigurator](#) class method), 16

`connection_ready` (*aries_cloudagent.messaging.request_context.RequestContext* attribute), 56
`connection_record` (*aries_cloudagent.messaging.request_context.RequestContext* attribute), 56
`ContextBuilder` (class in `create_session` (*aries_cloudagent.transport.inbound.manager.InboundManager*)), 69
`copy` (*aries_cloudagent.config.base.BaseInjector* method), 11
`copy` (*aries_cloudagent.config.base.BaseSettings* method), 11
`copy` (*aries_cloudagent.config.injection_context.InjectionContext* method), 13
`copy` (*aries_cloudagent.config.injector.Injector* method), 15
`copy` (*aries_cloudagent.config.settings.Settings* method), 17
`copy` (*aries_cloudagent.messaging.decorators.base.BaseDecorator* method), 32
`copy` (*aries_cloudagent.messaging.request_context.RequestContext* attribute), 56
`create` (*aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator* class method), 35
`create` (*aries_cloudagent.wallet.indy.IndyWallet* method), 86
`create_credential` (*aries_cloudagent.issuer.indy.IndyIssuer* method), 20
`create_credential_offer` (*aries_cloudagent.issuer.indy.IndyIssuer* method), 20
`create_credential_request` (*aries_cloudagent.holder.indy.IndyHolder* method), 18
`create_keypair` (in module *aries_cloudagent.wallet.crypto*), 82
`create_local_did` (*aries_cloudagent.wallet.base.BaseWallet* method), 76
`create_local_did` (*aries_cloudagent.wallet.basic.BasicWallet* method), 79
`create_local_did` (*aries_cloudagent.wallet.indy.IndyWallet* method), 86
`create_outbound` (*aries_cloudagent.messaging.responder.BaseResponder* method), 56
`create_pool_config` (*aries_cloudagent.ledger.indy.IndyLedger* method), 23
`create_presentation` (*aries_cloudagent.holder.indy.IndyHolder* method), 19
`create_session` (*aries_cloudagent.transport.inbound.manager.InboundManager* method), 69
`create_signing_key` (*aries_cloudagent.wallet.base.BaseWallet* method), 76
`create_signing_key` (*aries_cloudagent.wallet.basic.BasicWallet* method), 79
`create_signing_key` (*aries_cloudagent.wallet.indy.IndyWallet* method), 86
`created` (*aries_cloudagent.wallet.base.BaseWallet* attribute), 76
`created` (*aries_cloudagent.wallet.basic.BasicWallet* attribute), 79
`created` (*aries_cloudagent.wallet.indy.IndyWallet* attribute), 86
`created_at` (*aries_cloudagent.messaging.models.base_record.BaseRecord* attribute), 46
`credential_definition_id2schema_id` (*aries_cloudagent.ledger.indy.IndyLedger* method), 23
`credential_definitions_created` (in module *aries_cloudagent.messaging.credential_definitions.routes*), 30
`credential_definitions_get_credential_definition` (in module *aries_cloudagent.messaging.credential_definitions.routes*), 30
`credential_definitions_send_credential_definition` (in module *aries_cloudagent.messaging.credential_definitions.routes*), 31
`CredentialDefinitionGetResultsSchema` (class in *aries_cloudagent.messaging.credential_definitions.routes*), 28
`CredentialDefinitionSchema` (class in *aries_cloudagent.messaging.credential_definitions.routes*), 29
`CredentialDefinitionsCreatedResultsSchema` (class in *aries_cloudagent.messaging.credential_definitions.routes*), 30
`CredentialDefinitionSendRequestSchema` (class in *aries_cloudagent.messaging.credential_definitions.routes*), 29
`CredentialDefinitionSendResultsSchema` (class in *aries_cloudagent.messaging.credential_definitions.routes*), 29
`D`
`datetime_now` (in module

`aries_cloudagent.messaging.util`), 58
`datetime_to_str()` (in module `aries_cloudagent.messaging.util`), 58
`DebugGroup` (class in `aries_cloudagent.config.argparse`), 9
`decode()` (`aries_cloudagent.messaging.decorators.signature_decorator`), 53
`decode_pack_message()` (in module `aries_cloudagent.wallet.crypto`), 82
`decode_pack_message_outer()` (in module `aries_cloudagent.wallet.crypto`), 83
`decode_pack_message_payload()` (in module `aries_cloudagent.wallet.crypto`), 83
`DecoratorError`, 33
`DecoratorSet` (class in `aries_cloudagent.messaging.decorators.default`), 33
`decrypt_plaintext()` (in module `aries_cloudagent.wallet.crypto`), 83
`default_endpoint` (`aries_cloudagent.messaging.request_context.RequestContext` attribute), 56
`DEFAULT_FRESHNESS` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 85
`DEFAULT_KEY` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 85
`DEFAULT_KEY_DERIVATION` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 85
`default_label` (`aries_cloudagent.messaging.request_context.RequestContext` attribute), 56
`DEFAULT_NAME` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 85
`DEFAULT_STORAGE_TYPE` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 85
`DefaultContextBuilder` (class in `aries_cloudagent.config.default_context`), 13
`delay_milli` (`aries_cloudagent.messaging.decorators.timing_decorator.TimingDecorator` attribute), 39
`delete_credential()` (`aries_cloudagent.holder.indy.IndyHolder` method), 19
`delete_record()` (`aries_cloudagent.messaging.models.base_record.BaseRecord` method), 44
`delete_record()` (`aries_cloudagent.storage.base.BaseStorage` method), 59
`delete_record()` (`aries_cloudagent.storage.basic.BasicStorage` method), 61
`delete_record()` (`aries_cloudagent.storage.indy.IndyStorage` method), 63
`delete_record_tags()` (`aries_cloudagent.storage.base.BaseStorage` method), 59
`delete_record_tags()` (`aries_cloudagent.storage.basic.BasicStorage` method), 61
`delete_record_tags()` (`aries_cloudagent.storage.indy.IndyStorage` method), 64
`deliver_queued_message()` (`aries_cloudagent.transport.outbound.manager.OutboundTransportManager` method), 72
`deserialize()` (`aries_cloudagent.messaging.models.base.BaseModel` class method), 41
`did` (`aries_cloudagent.wallet.base.DIDInfo` attribute), 78
`did_to_nym()` (`aries_cloudagent.ledger.base.BaseLedger` method), 21
`DIDInfo` (class in `aries_cloudagent.wallet.base`), 78
`DIDListSchema` (class in `aries_cloudagent.wallet.routes`), 89
`DIDContext` (class in `aries_cloudagent.wallet.routes`), 90
`DIDSchema` (class in `aries_cloudagent.wallet.routes`), 90
`dispatch_complete()` (`aries_cloudagent.transport.inbound.manager.InboundTransportManager` method), 69
`done` (`aries_cloudagent.cache.base.CacheKeyLock` attribute), 6
`dump_decorators()` (`aries_cloudagent.messaging.agent_message.AgentMessageSchema` method), 54

E

`encode()` (in module `aries_cloudagent.issuer.util`), 21
`encode_pack_message()` (in module `aries_cloudagent.wallet.crypto`), 83
`encode_queued_message()` (`aries_cloudagent.transport.outbound.manager.OutboundTransportManager` method), 73
`encrypt_plaintext()` (in module `aries_cloudagent.wallet.crypto`), 83
`enqueue_message()` (`aries_cloudagent.transport.outbound.manager.OutboundTransportManager` method), 73
`enqueue_record_lock()` (`aries_cloudagent.transport.outbound.manager.OutboundTransportManager` method), 73
`epoch_to_str()` (in module `aries_cloudagent.messaging.util`), 58
`error_code` (`aries_cloudagent.messaging.error.MessageParseError` attribute), 55
`error_code` (`aries_cloudagent.messaging.error.MessagePrepareError` attribute), 55

`execute()` (in module `aries_cloudagent.commands.help`), 8
`execute()` (in module `aries_cloudagent.commands.provision`), 8
`execute()` (in module `aries_cloudagent.commands.start`), 8
`expires_time()` (`aries_cloudagent.messaging.decorators.timing_decorator.timing_decorator` attribute), 39
`extend()` (`aries_cloudagent.config.base.BaseSettings` method), 11
`extend()` (`aries_cloudagent.config.settings.Settings` method), 17
`extract_decorators()` (`aries_cloudagent.messaging.agent_message.AgentMessage` method), 54
`extract_decorators()` (`aries_cloudagent.messaging.decorators.base.BaseDecoratorSet` method), 32
`extract_pack_recipients()` (in module `aries_cloudagent.wallet.crypto`), 84
`extract_payload_key()` (in module `aries_cloudagent.wallet.crypto`), 84

F

`fetch()` (`aries_cloudagent.storage.base.BaseStorageRecordSearch` method), 60
`fetch()` (`aries_cloudagent.storage.basic.BasicStorageRecordSearch` method), 62
`fetch()` (`aries_cloudagent.storage.indy.IndyStorageRecordSearch` method), 65
`fetch_all()` (`aries_cloudagent.storage.base.BaseStorageRecordSearch` method), 60
`fetch_credential_definition()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 23
`fetch_genesis_transactions()` (in module `aries_cloudagent.config.ledger`), 15
`fetch_schema_by_id()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 24
`fetch_schema_by_seq_no()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 24
`fetch_single()` (`aries_cloudagent.storage.base.BaseStorageRecordSearch` method), 60
`fetch_txn_author_agreement()` (`aries_cloudagent.ledger.base.BaseLedger` method), 21
`fetch_txn_author_agreement()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 24
`field()` (`aries_cloudagent.messaging.decorators.base.BaseDecoratorSet` method), 32

G

`Group` (class in `aries_cloudagent.config.argsparse`), 9
`get_wallet_key()` (`aries_cloudagent.wallet.indy.IndyWallet` class method), 86
`get()` (`aries_cloudagent.cache.base.BaseCache` method), 6
`get()` (`aries_cloudagent.cache.basic.BasicCache` method), 7
`get_bool()` (`aries_cloudagent.config.base.BaseSettings` method), 12
`get_cached_key()` (`aries_cloudagent.messaging.models.base_record.BaseRecord` class method), 44
`get_credential()` (`aries_cloudagent.holder.indy.IndyHolder` method), 19
`get_credential_definition()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 24
`get_credential_definition_tag_policy()` (`aries_cloudagent.wallet.indy.IndyWallet` method), 86
`get_credentials()` (`aries_cloudagent.holder.indy.IndyHolder` method), 19
`get_credentials_for_presentation_request_by_referent()` (`aries_cloudagent.holder.indy.IndyHolder` method), 19
`get_for_did()` (`aries_cloudagent.ledger.base.BaseLedger` method), 21

```

get_endpoint_for_did()
    (aries_cloudagent.ledger.indy.IndyLedger
     method), 24
get_indy_storage()
    (aries_cloudagent.ledger.indy.IndyLedger
     method), 24
get_int() (aries_cloudagent.config.base.BaseSettings
          method), 12
get_key_for_did()
    (aries_cloudagent.ledger.base.BaseLedger
     method), 21
get_key_for_did()
    (aries_cloudagent.ledger.indy.IndyLedger
     method), 24
get_latest_txn_author_acceptance()
    (aries_cloudagent.ledger.base.BaseLedger
     method), 21
get_latest_txn_author_acceptance()
    (aries_cloudagent.ledger.indy.IndyLedger
     method), 24
get_local_did() (aries_cloudagent.wallet.base.BaseWallet
                method), 76
get_local_did() (aries_cloudagent.wallet.basic.BasicWallet
                method), 79
get_local_did() (aries_cloudagent.wallet.indy.IndyWallet
                method), 86
get_local_did_for_verkey()
    (aries_cloudagent.wallet.base.BaseWallet
     method), 77
get_local_did_for_verkey()
    (aries_cloudagent.wallet.basic.BasicWallet
     method), 79
get_local_did_for_verkey()
    (aries_cloudagent.wallet.indy.IndyWallet
     method), 87
get_local_dids() (aries_cloudagent.wallet.base.BaseWallet
                 method), 77
get_local_dids() (aries_cloudagent.wallet.basic.BasicWallet
                 method), 80
get_local_dids() (aries_cloudagent.wallet.indy.IndyWallet
                 method), 87
get_mime_type() (aries_cloudagent.holder.indy.IndyHolder
                method), 19
get_provider() (aries_cloudagent.config.injector.Injector
               method), 15
get_public_did() (aries_cloudagent.wallet.base.BaseWallet
                 method), 77
get_record() (aries_cloudagent.storage.base.BaseStorage
             method), 59
get_record() (aries_cloudagent.storage.basic.BasicStorage
             method), 61
get_record() (aries_cloudagent.storage.indy.IndyStorage
             method), 64
get_registered() (aries_cloudagent.config.argsparse.group
                 class method), 11
get_registered_transport_for_scheme()
    (aries_cloudagent.transport.outbound.manager.OutboundTransportManager
     method), 73
get_running_transport_for_endpoint()
    (aries_cloudagent.transport.outbound.manager.OutboundTransportManager
     method), 73
get_running_transport_for_scheme()
    (aries_cloudagent.transport.outbound.manager.OutboundTransportManager
     method), 73
get_schema() (aries_cloudagent.ledger.indy.IndyLedger
             method), 24
get_settings() (aries_cloudagent.config.argsparse.AdminGroup
               method), 9
get_settings() (aries_cloudagent.config.argsparse.ArgumentGroup
               method), 9
get_settings() (aries_cloudagent.config.argsparse.DebugGroup
               method), 9
get_settings() (aries_cloudagent.config.argsparse.GeneralGroup
               method), 9
get_settings() (aries_cloudagent.config.argsparse.LedgerGroup
               method), 10
get_settings() (aries_cloudagent.config.argsparse.LoggingGroup
               method), 10
get_settings() (aries_cloudagent.config.argsparse.ProtocolGroup
               method), 10
get_settings() (aries_cloudagent.config.argsparse.TransportGroup
               method), 10
get_settings() (aries_cloudagent.config.argsparse.WalletGroup
               method), 11
get_signature() (aries_cloudagent.messaging.agent_message.AgentMessage
                method), 52
get_signing_key()
    (aries_cloudagent.wallet.base.BaseWallet
     method), 77
get_signing_key()
    (aries_cloudagent.wallet.basic.BasicWallet
     method), 80
get_signing_key()
    (aries_cloudagent.wallet.indy.IndyWallet
     method), 87
get_str() (aries_cloudagent.config.base.BaseSettings
          method), 12
get_tag_map() (aries_cloudagent.messaging.models.base_record.BaseRecord
              class method), 44
get_transport_instance()
    (aries_cloudagent.transport.inbound.manager.InboundTransportManager
     method), 69
get_transport_instance()
    (aries_cloudagent.transport.outbound.manager.OutboundTransportManager
     method), 73
get_txn_author_agreement()
    (aries_cloudagent.ledger.base.BaseLedger
     method), 22

```

[get_txn_author_agreement\(\)](#) ([aries_cloudagent.ledger.indy.IndyLedger](#) [method](#)), 24
[get_value\(\)](#) ([aries_cloudagent.config.base.BaseSettings](#) [method](#)), 12
[get_value\(\)](#) ([aries_cloudagent.config.settings.Settings](#) [method](#)), 17
[GetTagPolicyResultSchema](#) ([class](#) [in](#) [aries_cloudagent.wallet.routes](#)), 90
[group](#) ([class](#) [in](#) [aries_cloudagent.config.argparse](#)), 11
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.AdminGroup](#) [attribute](#)), 9
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.ArgumentGroup](#) [attribute](#)), 9
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.DebugGroup](#) [attribute](#)), 9
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.GeneralGroup](#) [attribute](#)), 9
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.LedgerGroup](#) [attribute](#)), 10
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.LoggingGroup](#) [attribute](#)), 10
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.ProtocolGroup](#) [attribute](#)), 10
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.TransportGroup](#) [attribute](#)), 10
[GROUP_NAME](#) ([aries_cloudagent.config.argparse.WalletGroup](#) [attribute](#)), 10
H
[handle](#) ([aries_cloudagent.storage.base.BaseStorageRecordSearch](#) [attribute](#)), 60
[handle](#) ([aries_cloudagent.storage.indy.IndyStorageRecordSearch](#) [attribute](#)), 65
[handle](#) ([aries_cloudagent.wallet.base.BaseWallet](#) [attribute](#)), 77
[handle](#) ([aries_cloudagent.wallet.indy.IndyWallet](#) [attribute](#)), 87
[handle\(\)](#) ([aries_cloudagent.messaging.base_handler.BaseHandler](#) [method](#)), 54
[handle_message\(\)](#) ([aries_cloudagent.transport.outbound.base.BaseOutboundTransport](#) [method](#)), 71
[handle_message\(\)](#) ([aries_cloudagent.transport.outbound.http.HttpTransport](#) [method](#)), 72
[handle_message\(\)](#) ([aries_cloudagent.transport.outbound.ws.WsTransport](#) [method](#)), 75
[Handler](#) ([aries_cloudagent.messaging.agent_message.AgentMessage](#) [attribute](#)), 52
[handler_class](#) ([aries_cloudagent.messaging.agent_message.AgentMessageMeta](#) [attribute](#)), 52
[HandlerException](#), 55
[has_field\(\)](#) ([aries_cloudagent.messaging.decorators.base.BaseDecorator](#) [method](#)), 32
[host](#) ([aries_cloudagent.transport.inbound.base.InboundTransportConfiguration](#) [attribute](#)), 67
[HttpTransport](#) ([class](#) [in](#) [aries_cloudagent.transport.inbound.http](#)), 68
[HttpTransport](#) ([class](#) [in](#) [aries_cloudagent.transport.outbound.http](#)), 72
[InboundMessageHandler](#) ([class](#) [in](#) [aries_cloudagent.transport.inbound.http.HttpTransport](#) [method](#)), 68
[InboundMessageHandler](#) ([class](#) [in](#) [aries_cloudagent.transport.inbound.ws.WsTransport](#) [method](#)), 70
[InboundTransportConfiguration](#) ([class](#) [in](#) [aries_cloudagent.transport.inbound.base](#)), 67
[InboundTransportError](#), 67
[InboundTransportManager](#) ([class](#) [in](#) [aries_cloudagent.transport.inbound.manager](#)), 69
[InboundTransportRegistrationError](#), 68
[InboundTransportSetupError](#), 68
[IndyErrorHandler](#) ([class](#) [in](#) [aries_cloudagent.ledger.indy](#)), 23
[IndyHolder](#) ([class](#) [in](#) [aries_cloudagent.holder.indy](#)), 18
[IndyIssuer](#) ([class](#) [in](#) [aries_cloudagent.issuer.indy](#)), 20
[IndyLedger](#) ([class](#) [in](#) [aries_cloudagent.ledger.indy](#)), 23
[IndyStorage](#) ([class](#) [in](#) [aries_cloudagent.storage.indy](#)), 63
[IndyStorageRecordSearch](#) ([class](#) [in](#) [aries_cloudagent.storage.indy](#)), 65
[IndyVerifier](#) ([class](#) [in](#) [aries_cloudagent.verifier.indy](#)), 75
[IndyWallet](#) ([class](#) [in](#) [aries_cloudagent.wallet.indy](#)), 85
[init_argument_parser\(\)](#) ([in](#) [aries_cloudagent.commands.provision](#) [module](#)), 8
[init_argument_parser\(\)](#) ([in](#) [aries_cloudagent.commands.start](#) [module](#)), 8
[inject\(\)](#) ([aries_cloudagent.config.base.BaseInjector](#) [method](#)), 11
[inject\(\)](#) ([aries_cloudagent.config.injection_context.InjectionContext](#) [method](#)), 15
[inject\(\)](#) ([aries_cloudagent.config.injector.Injector](#) [method](#)), 13

[InjectionContextError](#), 14
[injector](#) (*aries_cloudagent.config.injection_context.InjectionContext* attribute), 14
[injector](#) (*aries_cloudagent.config.injection_context.Scope* attribute), 14
[Injector](#) (class in *aries_cloudagent.config.injector*), 15
[injector_for_scope](#) (*aries_cloudagent.config.injection_context.InjectionContext* method), 14
[InjectorError](#), 12
[InstanceProvider](#) (class in *aries_cloudagent.config.provider*), 17
[invite_message_handler](#) (*aries_cloudagent.transport.inbound.http.HttpTransport* method), 68
[IssuerError](#), 21

K

[KEY_DERIVATION_ARGON2I_INT](#) (*aries_cloudagent.wallet.indy.IndyWallet* attribute), 85
[KEY_DERIVATION_ARGON2I_MOD](#) (*aries_cloudagent.wallet.indy.IndyWallet* attribute), 85
[KEY_DERIVATION_RAW](#) (*aries_cloudagent.wallet.indy.IndyWallet* attribute), 85
[KeyInfo](#) (class in *aries_cloudagent.wallet.base*), 78

L

[LEDGER_CLASSES](#) (*aries_cloudagent.ledger.provider.LedgerProvider* attribute), 25
[ledger_config](#) (in module *aries_cloudagent.config.ledger*), 15
[LEDGER_TYPE](#) (*aries_cloudagent.ledger.base.BaseLedger* attribute), 21
[LEDGER_TYPE](#) (*aries_cloudagent.ledger.indy.IndyLedger* attribute), 23
[LedgerConfigError](#), 22
[LedgerError](#), 22
[LedgerGroup](#) (class in *aries_cloudagent.config.argparse*), 9
[LedgerProvider](#) (class in *aries_cloudagent.ledger.provider*), 25
[LedgerTransactionError](#), 22
[load_argument_groups](#) (in module *aries_cloudagent.config.argparse*), 11
[load_command](#) (in module *aries_cloudagent.commands*), 7
[load_decorator](#) (*aries_cloudagent.messaging.decorators.base.BaseDecorator* method), 32
[load_plugins](#) (*aries_cloudagent.config.default_context.DefaultContextBuilder* method), 13
[load_resource](#) (in module *aries_cloudagent.config.logging*), 16
[locale](#) (*aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecorator* attribute), 34
[localizable](#) (*aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecorator* attribute), 34
[LocalizationDecorator](#) (class in *aries_cloudagent.messaging.decorators.localization_decorator*), 33
[LocalizationDecorator.Meta](#) (class in *aries_cloudagent.messaging.decorators.localization_decorator*), 34
[LocalizationDecoratorSchema](#) (class in *aries_cloudagent.messaging.decorators.localization_decorator*), 34
[LocalizationDecoratorSchema.Meta](#) (class in *aries_cloudagent.messaging.decorators.localization_decorator*), 34
[log_state](#) (*aries_cloudagent.messaging.models.base_record.BaseRecord* class method), 44
[LOG_STATE_FLAG](#) (*aries_cloudagent.messaging.models.base_record.BaseRecord* attribute), 43
[LoggingConfigurator](#) (class in *aries_cloudagent.config.logging*), 15
[LoggingGroup](#) (class in *aries_cloudagent.config.argparse*), 10

M

[make_application](#) (*aries_cloudagent.admin.server.AdminServer* method), 5
[make_application](#) (*aries_cloudagent.transport.inbound.http.HttpTransport* method), 68
[make_application](#) (*aries_cloudagent.transport.inbound.ws.WsTransport* method), 70
[make_model](#) (*aries_cloudagent.messaging.models.base.BaseModelSchema* method), 42
[master_secret_id](#) (*aries_cloudagent.wallet.indy.IndyWallet* attribute), 87
[match_post_filter](#) (in module *aries_cloudagent.messaging.models.base_record*), 46
[max_message_size](#) (*aries_cloudagent.transport.inbound.base.BaseInboundTransport* attribute), 67
[message](#) (*aries_cloudagent.messaging.request_context.RequestContext* attribute), 56
[message_receipt](#) (*aries_cloudagent.messaging.request_context.RequestContext* attribute), 56
[message_base.BaseDecorator](#) (*aries_cloudagent.messaging.agent_message.AgentMessage* attribute), 52
[MessagePrepareError](#), 55

<code>metadata</code>	<code>(aries_cloudagent.wallet.base.DIDInfo attribute), 78</code>	<code>open()</code>	<code>(aries_cloudagent.wallet.base.BaseWallet method), 77</code>
<code>metadata</code>	<code>(aries_cloudagent.wallet.base.KeyInfo attribute), 78</code>	<code>open()</code>	<code>(aries_cloudagent.wallet.basic.BasicWallet method), 80</code>
<code>MockResponder</code>	<code>(class in aries_cloudagent.messaging.responder), 57</code>	<code>open()</code>	<code>(aries_cloudagent.wallet.indy.IndyWallet method), 87</code>
<code>Model</code>	<code>(aries_cloudagent.messaging.models.base.BaseModelSchema attribute), 60</code>	<code>opened</code>	<code>(aries_cloudagent.storage.base.BaseStorageRecordSearch attribute), 60</code>
<code>model_class</code>	<code>(aries_cloudagent.messaging.agent_message.AgentMessageSchema attribute), 54</code>	<code>opened</code>	<code>(aries_cloudagent.storage.basics.BasicStorageRecordSearch attribute), 60</code>
<code>model_class</code>	<code>(aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecoratorSchema.Meta attribute), 34</code>	<code>opened</code>	<code>(aries_cloudagent.storage.indy.IndyStorageRecordSearch attribute), 90</code>
<code>model_class</code>	<code>(aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecoratorSchema.Meta attribute), 35</code>	<code>opened</code>	<code>(aries_cloudagent.wallet.base.BaseWallet attribute), 77</code>
<code>model_class</code>	<code>(aries_cloudagent.messaging.decorators.thread_decorator.ThreadDecoratorSchema.Meta attribute), 37</code>	<code>opened</code>	<code>(aries_cloudagent.wallet.basics.BasicWallet attribute), 80</code>
<code>model_class</code>	<code>(aries_cloudagent.messaging.decorators.timing_decorator.TimingDecoratorSchema.Meta attribute), 38</code>	<code>option()</code>	<code>(aries_cloudagent.storage.base.BaseStorageRecordSearch attribute), 60</code>
<code>model_class</code>	<code>(aries_cloudagent.messaging.decorators.transport_decorator.TransportDecoratorSchema.Meta attribute), 40</code>	<code>options</code>	<code>(aries_cloudagent.storage.base.BaseStorageRecordSearch attribute), 60</code>
<code>model_class</code>	<code>(aries_cloudagent.messaging.models.base.BaseModelSchema.Meta attribute), 42</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.credential_definitions.routes.CredentialsRoutes attribute), 30</code>
<code>models</code>	<code>(aries_cloudagent.messaging.decorators.base.BaseDecoratorSchema attribute), 29</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.credential_definitions.routes.CredentialsRoutes attribute), 30</code>
<code>module</code>	<code>(aries_cloudagent.transport.inbound.base.InboundTransportConfiguration attribute), 67</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.credential_definitions.routes.CredentialsRoutes attribute), 30</code>
N		<code>opts</code>	<code>(aries_cloudagent.messaging.credential_definitions.routes.CredentialsRoutes attribute), 29</code>
<code>name</code>	<code>(aries_cloudagent.config.injection_context.Scope attribute), 14</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.credential_definitions.routes.CredentialsRoutes attribute), 30</code>
<code>name</code>	<code>(aries_cloudagent.wallet.base.BaseWallet attribute), 77</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.schemas.routes.SchemaGetResultsSchema attribute), 49</code>
<code>name</code>	<code>(aries_cloudagent.wallet.basic.BasicWallet attribute), 80</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.schemas.routes.SchemaSchema attribute), 49</code>
<code>name</code>	<code>(aries_cloudagent.wallet.indy.IndyWallet attribute), 87</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.schemas.routes.SchemasCreatedResultsSchema attribute), 50</code>
<code>nym_to_did()</code>	<code>(aries_cloudagent.ledger.base.BaseLedger method), 22</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.schemas.routes.SchemaSendRequestSchema attribute), 50</code>
<code>nym_to_did()</code>	<code>(aries_cloudagent.ledger.indy.IndyLedger method), 24</code>	<code>opts</code>	<code>(aries_cloudagent.messaging.schemas.routes.SchemaSendResultsSchema attribute), 50</code>
O		<code>opts</code>	<code>(aries_cloudagent.wallet.crypto.PackMessageSchema attribute), 81</code>
<code>on_startup()</code>	<code>(aries_cloudagent.admin.server.AdminServer method), 5</code>	<code>opts</code>	<code>(aries_cloudagent.wallet.crypto.PackRecipientHeaderSchema attribute), 82</code>
<code>open()</code>	<code>(aries_cloudagent.ledger.indy.IndyLedger method), 24</code>	<code>opts</code>	<code>(aries_cloudagent.wallet.crypto.PackRecipientSchema attribute), 82</code>
<code>open()</code>	<code>(aries_cloudagent.storage.base.BaseStorageRecordSearch method), 60</code>	<code>opts</code>	<code>(aries_cloudagent.wallet.crypto.PackRecipientsSchema attribute), 82</code>
<code>open()</code>	<code>(aries_cloudagent.storage.basics.BasicStorageRecordSearch method), 62</code>	<code>opts</code>	<code>(aries_cloudagent.wallet.routes.DIDListSchema attribute), 90</code>
<code>open()</code>	<code>(aries_cloudagent.storage.indy.IndyStorageRecordSearch method), 65</code>	<code>opts</code>	<code>(aries_cloudagent.wallet.routes.DIDResultSchema attribute), 90</code>

[opts \(aries_cloudagent.wallet.routes.DIDSchema attribute\), 90](#)
[opts \(aries_cloudagent.wallet.routes.GetTagPolicyResultSchema attribute\), 90](#)
[opts \(aries_cloudagent.wallet.routes.SetTagPolicyRequestSchema attribute\), 90](#)
[ordered \(aries_cloudagent.messaging.models.base.BaseModelSchema attribute\), 42](#)
[out_time \(aries_cloudagent.messaging.decorators.timing_decorator attribute\), 39](#)
[OutboundDeliveryError, 71](#)
[OutboundTransportError, 71](#)
[OutboundTransportManager \(class in aries_cloudagent.transport.outbound.manager\), 72](#)
[OutboundTransportRegistrationError, 72](#)

P

[pack_message \(\) \(aries_cloudagent.wallet.base.BaseWallet method\), 77](#)
[pack_message \(\) \(aries_cloudagent.wallet.basic.BasicWallet method\), 80](#)
[pack_message \(\) \(aries_cloudagent.wallet.indy.IndyWallet method\), 87](#)
[PackMessageSchema \(class in aries_cloudagent.wallet.crypto\), 81](#)
[PackRecipientHeaderSchema \(class in aries_cloudagent.wallet.crypto\), 81](#)
[PackRecipientSchema \(class in aries_cloudagent.wallet.crypto\), 82](#)
[PackRecipientsSchema \(class in aries_cloudagent.wallet.crypto\), 82](#)
[pad \(\) \(in module aries_cloudagent.wallet.util\), 92](#)
[page_size \(aries_cloudagent.storage.base.BaseStorageRecord attribute\), 60](#)
[parent \(aries_cloudagent.cache.base.CacheKeyLock attribute\), 7](#)
[perform_encode \(\) \(aries_cloudagent.transport.outbound.manager.OutboundTransportManager method\), 73](#)
[plugins_handler \(\) \(aries_cloudagent.admin.server.AdminServer method\), 5](#)
[populate_decorators \(\) \(aries_cloudagent.messaging.agent_message.AgentMessageSchema method\), 54](#)
[port \(aries_cloudagent.transport.inbound.base.InboundTransportConfiguration attribute\), 67](#)
[post_save \(\) \(aries_cloudagent.messaging.models.base_record.BaseRecord method\), 44](#)
[prefix \(aries_cloudagent.messaging.decorators.base.BaseDecorator attribute\), 32](#)
[prefix_tag_filter \(\) \(aries_cloudagent.messaging.models.base_record.BaseRecord class method\), 44](#)

Q

[queued_message_count \(aries_cloudagent.messaging.decorators.transport_decorator.Transaction attribute\), 40](#)
[QueuedOutboundMessage \(class in aries_cloudagent.transport.outbound.manager\), 74](#)

R

[random_seed \(\) \(in module aries_cloudagent.wallet.crypto\), 84](#)
[received_orders \(aries_cloudagent.messaging.decorators.thread_decorator attribute\), 36](#)
[received_orders \(aries_cloudagent.messaging.decorators.thread_decorator attribute\), 37](#)

RECORD_ID_NAME (aries_cloudagent.messaging.models.base_record.BaseRecord), 43
record_id_metadata (aries_cloudagent.wallet.base.BaseWallet), 77
record_tags (aries_cloudagent.messaging.models.base_record.BaseRecord), 45
replace_local_did_metadata (aries_cloudagent.wallet.basic.BasicWallet), 80
RECORD_TYPE (aries_cloudagent.messaging.models.base_record.BaseRecord), 43
method), 80
RECORD_TYPE_MIME_TYPES (aries_cloudagent.wallet.indy.IndyWallet), 88
method), 88
record_value (aries_cloudagent.messaging.models.base_record.BaseRecord), 45
AgentMessageSchema), 54
redirect_handler (aries_cloudagent.admin.server.AdminServer), 5
replace_signing_key_metadata (aries_cloudagent.wallet.base.BaseWallet), 80
register (aries_cloudagent.transport.inbound.manager.InboundTransportManager), 69
replace_signing_key_metadata (aries_cloudagent.wallet.basic.BasicWallet), 80
register (aries_cloudagent.transport.outbound.manager.OutboundTransportManager), 73
method), 80
register (in module aries_cloudagent.messaging.credential_definitions.routes), (aries_cloudagent.wallet.indy.IndyWallet), 88
31
register (in module RequestContext), (class in aries_cloudagent.messaging.request_context), 55
50
register (in module resolve_class (in module aries_cloudagent.messaging.models.base), 90
aries_cloudagent.wallet.routes), 90
register_class (aries_cloudagent.transport.outbound.manager.OutboundTransportManager), 74
resolve_meta_property (in module aries_cloudagent.messaging.models.base), 43
register_nym (aries_cloudagent.ledger.base.BaseLedger), 22
method), 22
register_nym (aries_cloudagent.ledger.indy.IndyLedger), 24
method), 24
ResponderError, 57
result (aries_cloudagent.admin.server.AdminModulesSchema), 4
register_transport (aries_cloudagent.transport.inbound.manager.InboundTransportManager), 69
method), 69
CacheKeyLock (aries_cloudagent.cache.base.CacheKeyLock), 7
attribute), 7
release (aries_cloudagent.cache.base.BaseCache), 6
method), 6
retrieve_by_id (aries_cloudagent.messaging.models.base_record.BaseRecord), 45
class method), 45
release (aries_cloudagent.cache.base.CacheKeyLock), 7
method), 7
retrieve_by_tag_filter (aries_cloudagent.messaging.models.base_record.BaseRecord), 45
class method), 45
remove (aries_cloudagent.wallet.indy.IndyWallet), 88
method), 88
return_route (aries_cloudagent.messaging.decorators.transport_decorator.TransportDecorator), 40
Set (aries_cloudagent.messaging.decorators.base.BaseDecorator), 32
method), 32
return_route_thread (aries_cloudagent.messaging.decorators.transport_decorator.TransportDecorator), 40
attribute), 40
remove_model (aries_cloudagent.messaging.decorators.base.BaseDecorator), 33
method), 33
return_to_session (aries_cloudagent.transport.inbound.manager.InboundTransportManager), 70
method), 70
remove_skipped_values (aries_cloudagent.messaging.models.base.BaseModelSchema), 42
method), 42
return_undelivered (aries_cloudagent.transport.inbound.manager.InboundTransportManager), 70
method), 70
remove_webhook_target (aries_cloudagent.admin.base_server.BaseAdminServer), 3
method), 3
ROOT_SCOPE (aries_cloudagent.config.injection_context.InjectionContext), 13
attribute), 13
remove_webhook_target (aries_cloudagent.admin.server.AdminServer), 5
method), 5
run_command (in module

`aries_cloudagent.commands`), 8
`run_loop()` (in module `aries_cloudagent.commands.start`), 8
S
`save()` (`aries_cloudagent.messaging.models.base_record.BaseRecord` attribute), 45
`Schema` (`aries_cloudagent.messaging.models.base.BaseModel` attribute), 41
`schema_class` (`aries_cloudagent.messaging.agent_message.AgentMessage` attribute), 52
`schema_class` (`aries_cloudagent.messaging.decorators.localization_decorator.LocalizationDecorator` attribute), 34
`schema_class` (`aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator` attribute), 35
`schema_class` (`aries_cloudagent.messaging.decorators.thread_decorator.ThreadDecorator` attribute), 36
`schema_class` (`aries_cloudagent.messaging.decorators.timing_decorator.TimingDecorator` attribute), 38
`schema_class` (`aries_cloudagent.messaging.decorators.transport_decorator.TransportDecorator` attribute), 39
`schema_class` (`aries_cloudagent.messaging.models.base.BaseModel` attribute), 41
`SchemaGetResultsSchema` (class in `aries_cloudagent.messaging.schemas.routes`), 49
`schemas_created()` (in module `aries_cloudagent.messaging.schemas.routes`), 51
`schemas_get_schema()` (in module `aries_cloudagent.messaging.schemas.routes`), 51
`schemas_send_schema()` (in module `aries_cloudagent.messaging.schemas.routes`), 51
`SchemaSchema` (class in `aries_cloudagent.messaging.schemas.routes`), 49
`SchemasCreatedResultsSchema` (class in `aries_cloudagent.messaging.schemas.routes`), 50
`SchemaSendRequestSchema` (class in `aries_cloudagent.messaging.schemas.routes`), 49
`SchemaSendResultsSchema` (class in `aries_cloudagent.messaging.schemas.routes`), 50
`scheme` (`aries_cloudagent.transport.inbound.base.BaseInboundTransport` attribute), 67
`scheme` (`aries_cloudagent.transport.inbound.ws.WsTransport` attribute), 70
`schemes` (`aries_cloudagent.transport.outbound.http.HttpTransport` attribute), 72
`schemes` (`aries_cloudagent.transport.outbound.ws.WsTransport` attribute), 75
`Scope` (class in `aries_cloudagent.config.injection_context`), 14
`scope_name` (`aries_cloudagent.config.injection_context.InjectionContext` attribute), 14
`search_records()` (`aries_cloudagent.storage.base.BaseStorage` method), 59
`search_records()` (`aries_cloudagent.storage.basic.BasicStorage` method), 61
`search_records()` (`aries_cloudagent.storage.indy.IndyStorage` method), 64
`seed_to_did()` (in module `aries_cloudagent.wallet.crypto`), 84
`send()` (`aries_cloudagent.messaging.responder.BaseResponder` method), 57
`send()` (`aries_cloudagent.messaging.responder.MockResponder` method), 57
`send_credential_definition()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 24
`send_outbound()` (`aries_cloudagent.admin.server.AdminResponder` method), 4
`send_outbound()` (`aries_cloudagent.messaging.responder.BaseResponder` method), 57
`send_outbound()` (`aries_cloudagent.messaging.responder.MockResponder` method), 57
`send_reply()` (`aries_cloudagent.messaging.responder.BaseResponder` method), 57
`send_reply()` (`aries_cloudagent.messaging.responder.MockResponder` method), 57
`send_schema()` (`aries_cloudagent.ledger.indy.IndyLedger` method), 25
`send_webhook()` (`aries_cloudagent.admin.base_server.BaseAdminServer` method), 3
`send_webhook()` (`aries_cloudagent.admin.server.AdminResponder` method), 4
`send_webhook()` (`aries_cloudagent.admin.server.AdminServer` method), 5
`send_webhook()` (`aries_cloudagent.messaging.models.base_record.BaseRecord` method), 45
`send_webhook()` (`aries_cloudagent.messaging.responder.BaseResponder` method), 57
`send_webhook()` (`aries_cloudagent.messaging.responder.MockResponder` method), 57
`sender_order` (`aries_cloudagent.messaging.decorators.thread_decorator.ThreadDecorator` attribute), 36
`sender_order` (`aries_cloudagent.messaging.decorators.thread_decorator.ThreadDecorator` attribute), 37
`serialize()` (`aries_cloudagent.messaging.models.base.BaseModel` method), 42
`set()` (`aries_cloudagent.cache.base.BaseCache` method), 6
`set()` (`aries_cloudagent.cache.basic.BasicCache` method), 6

`method`), 7
`SignatureDecorator.Meta` (class in `aries_cloudagent.messaging.decorators.signature_decorator`), 34
`set_cached_key()` (`aries_cloudagent.messaging.models.base_record.BaseRecord` attribute), 46
`SignatureDecoratorSchema` (class in `aries_cloudagent.messaging.decorators.signature_decorator`), 35
`set_credential_definition_tag_policy()` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 88
`SignatureDecoratorSchema.Meta` (class in `aries_cloudagent.messaging.decorators.signature_decorator`), 35
`set_default()` (`aries_cloudagent.config.settings.Settings` attribute), 17
`set_public_did()` (`aries_cloudagent.wallet.base.BaseWallet` attribute), 78
`signed_fields` (`aries_cloudagent.messaging.agent_message.AgentMessage` attribute), 54
`set_result()` (`aries_cloudagent.cache.base.CacheKeyLock` attribute), 7
`signer` (`aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator` attribute), 35
`set_signature()` (`aries_cloudagent.messaging.agent_message.AgentMessage` attribute), 52
`skip_dump_only()` (`aries_cloudagent.messaging.models.base.BaseModelSchema` attribute), 42
`set_urlsafes_b64()` (in module `aries_cloudagent.wallet.util`), 92
`skip_values` (`aries_cloudagent.messaging.models.base.BaseModelSchema` attribute), 42
`set_value()` (`aries_cloudagent.config.settings.Settings` attribute), 18
`stale_time` (`aries_cloudagent.messaging.decorators.timing_decorator.TimingDecorator` attribute), 39
`SetTagPolicyRequestSchema` (class in `aries_cloudagent.wallet.routes`), 90
`start()` (`aries_cloudagent.admin.base_server.BaseAdminServer` attribute), 3
`settings` (`aries_cloudagent.config.injection_context.InjectionContext` attribute), 14
`start()` (`aries_cloudagent.admin.server.AdminServer` attribute), 5
`settings` (`aries_cloudagent.config.injector.Injector` attribute), 15
`start()` (`aries_cloudagent.transport.inbound.base.BaseInboundTransportManager` attribute), 67
`Settings` (class in `aries_cloudagent.config.settings`), 17
`start()` (`aries_cloudagent.transport.inbound.http.HttpTransportManager` attribute), 68
`SettingsError`, 12
`setup()` (`aries_cloudagent.transport.inbound.manager.InboundTransportManager` attribute), 70
`method`), 70
`setup()` (`aries_cloudagent.transport.outbound.manager.OutboundTransportManager` attribute), 74
`method`), 70
`shutdown_app()` (in module `aries_cloudagent.commands.start`), 8
`start()` (`aries_cloudagent.transport.outbound.base.BaseOutboundTransportManager` attribute), 71
`sig_data` (`aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator` attribute), 35
`method`), 72
`sign_field()` (`aries_cloudagent.messaging.agent_message.AgentMessage` attribute), 53
`method`), 74
`sign_message()` (`aries_cloudagent.wallet.base.BaseWallet` attribute), 78
`start()` (`aries_cloudagent.transport.outbound.ws.WsTransportManager` attribute), 75
`sign_message()` (`aries_cloudagent.wallet.basic.BasicWallet` attribute), 80
`start_app()` (in module `aries_cloudagent.commands.start`), 8
`sign_message()` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 88
`start_scope()` (`aries_cloudagent.config.injection_context.InjectionContext` attribute), 14
`sign_message()` (in module `aries_cloudagent.wallet.crypto`), 84
`start_transport()` (`aries_cloudagent.transport.inbound.manager.InboundTransportManager` attribute), 70
`sign_pk_from_sk()` (in module `aries_cloudagent.wallet.crypto`), 84
`start_transport()` (`aries_cloudagent.transport.outbound.manager.OutboundTransportManager` attribute), 74
`signature` (`aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator` attribute), 35
`method`), 74
`signature_type` (`aries_cloudagent.messaging.decorators.signature_decorator.SignatureDecorator` attribute), 35
`attribute`), 46
`SignatureDecorator` (class in `aries_cloudagent.messaging.decorators.signature_decorator`), 34
`STATE_DELIVER` (`aries_cloudagent.transport.outbound.manager.QueuedOutboundTransportManager` attribute), 74
`STATE_DONE` (`aries_cloudagent.transport.outbound.manager.QueuedOutboundTransportManager` attribute), 74

attribute), 74
 STATE_ENCODE (aries_cloudagent.transport.outbound.manager.QueueOutboundMessage (in module
 attribute), 74
 STATE_NEW (aries_cloudagent.transport.outbound.manager.QueueOutboundMessage (in module
 attribute), 74
 STATE_PENDING (aries_cloudagent.transport.outbound.manager.QueueOutboundMessage
 attribute), 74
 STATE_RETRY (aries_cloudagent.transport.outbound.manager.QueueOutboundMessage
 attribute), 74
 StatsProvider (class in T
 aries_cloudagent.config.provider), 17
 status_handler() (aries_cloudagent.admin.server.AdminServer method), 22
 method), 5
 status_reset_handler()
 (aries_cloudagent.admin.server.AdminServer
 method), 5
 stop() (aries_cloudagent.admin.base_server.BaseAdminServer
 method), 3
 stop() (aries_cloudagent.admin.server.AdminServer
 method), 5
 stop() (aries_cloudagent.transport.inbound.base.BaseInboundTransport
 method), 67
 stop() (aries_cloudagent.transport.inbound.http.HttpTransport
 method), 68
 stop() (aries_cloudagent.transport.inbound.manager.InboundTransportManager
 method), 70
 stop() (aries_cloudagent.transport.inbound.ws.WsTransport
 method), 70
 stop() (aries_cloudagent.transport.outbound.base.BaseOutboundTransport
 method), 71
 stop() (aries_cloudagent.transport.outbound.http.HttpTransport
 method), 72
 stop() (aries_cloudagent.transport.outbound.manager.OutboundTransportManager
 method), 74
 stop() (aries_cloudagent.transport.outbound.ws.WsTransport
 method), 75
 storage_record(aries_cloudagent.messaging.models.base_record.BaseRecord
 attribute), 46
 STORAGE_TYPES (aries_cloudagent.storage.provider.StorageProvider
 attribute), 66
 StorageDuplicateError, 63
 StorageError, 63
 StorageNotFoundError, 63
 StorageProvider (class in
 aries_cloudagent.storage.provider), 66
 StorageRecord (class in
 aries_cloudagent.storage.record), 66
 StorageSearchError, 63
 store(aries_cloudagent.storage.base.BaseStorageRecordSearch
 attribute), 60
 store_credential()
 (aries_cloudagent.holder.indy.IndyHolder
 method), 20
 str_to_b64() (in module
 aries_cloudagent.wallet.util), 92
 taa_digest() (aries_cloudagent.ledger.base.BaseLedger
 method), 25
 taa_digest() (aries_cloudagent.ledger.indy.IndyLedger
 method), 25
 taa_rough_timestamp()
 (aries_cloudagent.ledger.indy.IndyLedger
 method), 25
 TAG_NAMES (aries_cloudagent.messaging.models.base_record.BaseRecord
 attribute), 43
 tag_query(aries_cloudagent.storage.base.BaseStorageRecordSearch
 attribute), 60
 tags(aries_cloudagent.messaging.models.base_record.BaseRecord
 attribute), 46
 thid(aries_cloudagent.messaging.decorators.thread_decorator.ThreadDe
 attribute), 36
 thid(aries_cloudagent.messaging.decorators.thread_decorator.ThreadDe
 attribute), 37
 ThreadDecorator (class in
 aries_cloudagent.messaging.decorators.thread_decorator),
 36
 ThreadDecorator.Meta (class in
 aries_cloudagent.messaging.decorators.thread_decorator),
 36
 ThreadDecoratorSchema (class in
 aries_cloudagent.messaging.decorators.thread_decorator),
 36
 ThreadDecoratorSchema.Meta (class in
 aries_cloudagent.messaging.decorators.thread_decorator),
 36
 time_now() (in module
 aries_cloudagent.messaging.util), 58
 TimingDecorator (class in
 aries_cloudagent.messaging.decorators.timing_decorator),
 37
 TimingDecorator.Meta (class in
 aries_cloudagent.messaging.decorators.timing_decorator),
 38
 TimingDecoratorSchema (class in
 aries_cloudagent.messaging.decorators.timing_decorator),
 38
 TimingDecoratorSchema.Meta (class in
 aries_cloudagent.messaging.decorators.timing_decorator),
 38

[to_dict\(\)](#) ([aries_cloudagent.messaging.decorators.base.BaseDecoratorSet](#) [method](#)), 33
[to_json\(\)](#) ([aries_cloudagent.messaging.models.base.BaseModel](#) [method](#)), 62
[topic_filter\(\)](#) ([aries_cloudagent.admin.server.WebhookTarget](#) [attribute](#)), 5
[TransportDecorator](#) ([class](#) in [update_settings\(\)](#) [aries_cloudagent.messaging.decorators.transport_decorator.py](#) [method](#)), 13
[TransportDecorator.Meta](#) ([class](#) in [update_settings\(\)](#) [aries_cloudagent.messaging.decorators.transport_decorator.py](#) [method](#)), 14
[TransportDecoratorSchema](#) ([class](#) in [updated_at\(\)](#) [aries_cloudagent.messaging.models.base_record.BaseRecord](#) [attribute](#)), 46
[TransportDecoratorSchema.Meta](#) ([class](#) in [aries_cloudagent.messaging.decorators.transport_decorator.py](#) [method](#)), 39
[TransportGroup](#) ([class](#) in [value\(\)](#) [aries_cloudagent.config.argparse](#)), 10
[type](#) ([aries_cloudagent.wallet.base.BaseWallet](#) [attribute](#)), 78
[TYPE_ED25519SHA512](#) ([aries_cloudagent.messaging.decorators.signature_decorator.py](#) [attribute](#)), 35
[type_filter\(\)](#) ([aries_cloudagent.storage.base.BaseStorageRecordSearch](#) [attribute](#)), 61

U

[unpack_message\(\)](#) ([aries_cloudagent.wallet.base.BaseWallet](#) [method](#)), 78
[unpack_message\(\)](#) ([aries_cloudagent.wallet.basic.BasicWallet](#) [method](#)), 81
[unpack_message\(\)](#) ([aries_cloudagent.wallet.indy.IndyWallet](#) [method](#)), 89
[unpad\(\)](#) (in [module](#) [aries_cloudagent.wallet.util](#)), 92
[update_endpoint_for_did\(\)](#) ([aries_cloudagent.ledger.base.BaseLedger](#) [method](#)), 22
[update_endpoint_for_did\(\)](#) ([aries_cloudagent.ledger.indy.IndyLedger](#) [method](#)), 25
[update_record_tags\(\)](#) ([aries_cloudagent.storage.base.BaseStorage](#) [method](#)), 59
[update_record_tags\(\)](#) ([aries_cloudagent.storage.basic.BasicStorage](#) [method](#)), 62
[update_record_tags\(\)](#) ([aries_cloudagent.storage.indy.IndyStorage](#) [method](#)), 64
[update_record_value\(\)](#) ([aries_cloudagent.storage.base.BaseStorage](#) [method](#)), 59

V

[value\(\)](#) ([aries_cloudagent.messaging.models.base_record.BaseRecord](#) [attribute](#)), 46
[verify\(\)](#) ([aries_cloudagent.messaging.decorators.signature_decorator.py](#) [method](#)), 35
[verify_message\(\)](#) ([aries_cloudagent.wallet.base.BaseWallet](#) [method](#)), 78
[verify_message\(\)](#) ([aries_cloudagent.wallet.basic.BasicWallet](#) [method](#)), 81
[verify_message\(\)](#) ([aries_cloudagent.wallet.indy.IndyWallet](#) [method](#)), 89
[verify_presentation\(\)](#) ([aries_cloudagent.verifier.indy.IndyVerifier](#) [method](#)), 75
[verify_signatures\(\)](#) ([aries_cloudagent.messaging.agent_message.AgentMessage](#) [method](#)), 53
[verify_signed_field\(\)](#) ([aries_cloudagent.messaging.agent_message.AgentMessage](#) [method](#)), 53
[verify_signed_message\(\)](#) (in [module](#) [aries_cloudagent.wallet.crypto](#)), 84
[verkey](#) ([aries_cloudagent.wallet.base.DIDInfo](#) [attribute](#)), 78
[verkey](#) ([aries_cloudagent.wallet.base.KeyInfo](#) [attribute](#)), 79

W

[wait_until_time\(\)](#) ([aries_cloudagent.messaging.decorators.timing_decorator.py](#) [attribute](#)), 39
[wallet](#) ([aries_cloudagent.storage.indy.IndyStorage](#) [attribute](#)), 65
[wallet_config\(\)](#) (in [module](#) [aries_cloudagent.config.wallet](#)), 18
[wallet_create_did\(\)](#) (in [module](#) [aries_cloudagent.wallet.routes](#)), 91

`wallet_did_list()` (in module `aries_cloudagent.wallet.routes`), 91
`wallet_get_public_did()` (in module `aries_cloudagent.wallet.routes`), 91
`wallet_get_tagging_policy()` (in module `aries_cloudagent.wallet.routes`), 91
`wallet_set_public_did()` (in module `aries_cloudagent.wallet.routes`), 91
`wallet_set_tagging_policy()` (in module `aries_cloudagent.wallet.routes`), 91
`WALLET_TYPE` (`aries_cloudagent.wallet.base.BaseWallet` attribute), 76
`WALLET_TYPE` (`aries_cloudagent.wallet.basic.BasicWallet` attribute), 79
`WALLET_TYPE` (`aries_cloudagent.wallet.indy.IndyWallet` attribute), 85
`WALLET_TYPES` (`aries_cloudagent.wallet.provider.WalletProvider` attribute), 89
`WalletDuplicateError`, 85
`WalletError`, 85
`WalletGroup` (class in `aries_cloudagent.config.argparse`), 10
`WalletNotFoundError`, 85
`WalletProvider` (class in `aries_cloudagent.wallet.provider`), 89
`webhook_payload` (`aries_cloudagent.messaging.models.base_record.BaseRecord` attribute), 46
`WEBHOOK_TOPIC` (`aries_cloudagent.messaging.models.base_record.BaseRecord` attribute), 43
`webhook_topic` (`aries_cloudagent.messaging.models.base_record.BaseRecord` attribute), 46
`WebhookTarget` (class in `aries_cloudagent.admin.server`), 5
`websocket_handler()` (`aries_cloudagent.admin.server.AdminServer` method), 5
`wire_format` (`aries_cloudagent.transport.outbound.base.BaseOutboundTransport` attribute), 71
`wrap_error()` (`aries_cloudagent.ledger.indy.IndyErrorHandler` class method), 23
`WsTransport` (class in `aries_cloudagent.transport.inbound.ws`), 70
`WsTransport` (class in `aries_cloudagent.transport.outbound.ws`), 75