Agility Sync API Documentation

Release 22.1.0.0

The Agility Sync Authors

Jun 17, 2022

CONTENTS

1 Documentation **3**

1.1 User's guide 3
1.1.1 Introduction 3
1.2 Mapping 3
1.2.1 Introduction 3
1.2.2 AssetsManage — Assets Manage API 4
1.2.3 Fields — Fields API 6
1.2.4 Field — Field API 7
1.2.5 WebHook — WebHook API 10
1.3 Sync 12
1.3.1 Introduction 12
1.3.2 Payload — Payload API 12
1.3.3 Event — Event API 16
1.3.4 Inbound — Inbound API 17
1.3.5 Outbound — Outbound API 25
1.3.6 Attachment — Attachment API 31
1.4 Constants —Agility Sync Constants 34
1.5 Exceptions —Agility Sync Exceptions 37
1.6 External Plugin 40
1.6.1 Introduction 40
1.6.2 Plugin Configuration 41
1.6.3 FieldType Expression 44
1.6.4 Sync - Sample Sync Code 45
1.6.5 Mapping - Sample mapping Code 52

Python Module Index **57**

Agility Sync is an enterprise integration solution for integrating Digital.ai Agility with other ALM tools such as Jira, Azure Devops, and TeamForge. The sync between ALM system would happen through webhooks. It is a Python-based framework.

CHAPTER

ONE

DOCUMENTATION

1.1 User's guide

1.1.1 Introduction

Agility Sync is an enterprise integration solution for integrating Digital.ai Agility with other ALM tools such as Jira, Azure DevOps, and TeamForge. The sync between ALM system would happen through webhooks. It is a Python-based framework.

AgilitySync codebase comprises of two major components:

- Mapping (*AgilitySync.mapping*) is the Python package used to get the asset related information from ALM systems based on their configuration in the plugin system Agility Sync provides. It also is used for creating the mapping between any supported ALM system and Digitial.ai Agility to aid live syncing of data between the 2 ALM systems or to migrate from the other ALM system into Digital.ai Agility.
- Sync (AgilitySync.sync) is the Python package which is used for the actual syncing of data between ALM system thereby maintaining them both in sync. It operates based on webhooks received from the respective ALM systems on activities related to workitems happening in them.

1.2 Mapping

1.2.1 Introduction

Mapping is an Agility Sync Python package which comprises of classes AssetManage, Fields and Field. All plugins inherit these classes in a file mapping.py. The package exports the following packages which can be accessed by importing them from AgilitySync.mapping.

This package contains the functionality that aids in creating a mapping between Digital.ai Agility and another ALM system. It does all the things from displaying the instances, fetching the projects and any levels of hierarchies above that from the selected instances, fetching and displaying asset/workitem types from the selected projects and then provides the fields from the 2 ALM systems to be mapped for syncing. It manages the creation, update and delete of the data maps that contain this mapping between the ALM systems. Upon activation of a mapping which is an indication that the Agility Sync can start processing workitem syncs between the 2 mapped ALM systems, it also creates a webhook automatically on the respective ALM systems for receiving events relevant to activity involving the mapped assets.

1.2.2 AssetsManage — Assets Manage API

The AssetsManage class which will have to be implemented by all plugins with BaseAssetManage as base, handles fetching of all information from the respective ALM system like projects, and any elements above it in hierarchy and workitem types supported by the respective ALM systems.

BaseAssetManage Class

class AgilitySync.mapping.BaseAssetsManage(**args*, ***kwargs*)

This is the base class that the plugins of the ALM systems must inherit as *BaseAssetsManage* class. This class and its methods manage various assets/workitem types of a particular ALM system. It also contains some utility methods like one for testing connectivity to the respective ALM system.

Properties

BaseAssetsManage.query_params

The input value selected by customer.

It returns Dict which contains information about APIs.

BaseAssetsManage.instance_details

This property holds information about the specific instance of the plugin, in the form of a Dict.

BaseAssetsManage.instance_obj

This property holds the connection object to the specific instance of the plugin. Please refer *BaseAssetsManage.connect* for more info.

BaseAssetsManage.plugin_package_path

This property holds the path where the respective plugin is installed as a string.

Methods

BaseAssetsManage.fetch_projects() → List[Dict]

This method must be overridden by the plugin and must return the list of projects available in the particular instance of the plugin. The response expected is a List of Dict. Failure to override this method will result in an exception. Please see below for a sample response.

Sample Response:

```
[
    {
        "id": "unique ID of the project",

        "display_name": "Display name of the project",

        "project": "The value would be used during sync. Either "Id" or "display_name
→" can be used
    }
]
```

BaseAssetsManage.fetch_assets() → List[Dict]

This method must be overridden by the plugin to return the list of assets/workitem types in the particular instance of the plugin for the selected project(s) and any possible levels of hierarchies above it. The response expected is a List of Dict. Failure to override this method will result in an exception. Please see below for a sample response.

Sample Response:

```
[
    {
        "id": "unique ID of the asset",
        "display_name": "Display name of the asset",
        "asset": "The value would be used during sync. Either 'Id' or "display_name
↪" can be used
    }
]
```

BaseAssetsManage.fetch_sync_user() → Union[int, str]

This method must be overridden by the plugin to return an identification of the user like id, username, email etc. as used by the respective instance of the plugin, whose credentials will be used by Agility Sync to perform all syncing activities. The return of this method is used to compare and eliminate the events triggered by actions performed by this user so that cyclic events can be avoided. The response is expected as a str in case of login names, usernames, email addresses and may also be an int in case of user id. Failure to override this method by the plugin will result in an exception.

BaseAssetsManage.test_connection() → str

This method must be overridden by the plugin to run the necessary actions to make sure that the values provided for the configuration of an instance of the plugin are correct. This is usually done by trying to establish connection with that instance, using the details and credentials provided. The UI would use this method to test the connection to the instance of the plugin during the configuration of that instance or any point later. This method is called from plugin UI page while clicking test connection button. The response must be an str message indicating that the connection to the instance was successful. This message is displayed by the UI on the screen when the user clicks on the "Test Connection" button in the plugin instance configuration screen or instances list view screen.

The instance details needed to perform this operation are available in the *BaseAssetsManage.instance_details* property. An exception must be raised in case of failure to establish connection to the configured instance. Failure to override this method by the plugin will result in an exception.

BaseAssetsManage.connect() → Any

This method must be overridden by the plugin to return a connection object to the particular instance of the plugin. This connection object may later be accessed using BaseAssetsManage.instance_obj property wherever needed. Failure to override this method by the plugin will result in an exception.

BaseAssetsManage.is_instance_supported() → Tuple[bool, str]

The plugin may override this method to return a bool indicating whether the version of this instance of the plugin is supported or not, along with the minimum supported version for the plugin. So, if overridden this method is expected to return a Tuple of 2 values - a bool denoting whether the version the instance is on is supported or not, and a str indicating the minimum supported version for the plugin. The base method returns a Tuple of True and an empty string.

1.2.3 Fields — Fields API

The Fields class must be implemented by the respective plugins deriving from the BaseFields class. This class handles the fetching of fields of corresponding workitem types from the ALM systems.

BaseFields Class

class AgilitySync.mapping.BaseFields(**args*, ***kargs*)

This is the base class which will be inherited by the respective plugins of the ALM systems as Fields class to handle their respective fields of various workitem types supported.

Properties

BaseFields.asset_info

This property holds information about the selected asset in the form of a Dict. The selected asset will be one from the list of assets and their related information returned by the *BaseAssetsManage.fetch_assets* method.

BaseFields.projects

This property holds the list of selected projects as a List of Dict. The selected projects are from the list of projects returned by the *BaseAssetsManage.fetch_projects* method.

BaseFields.instance_obj

This property holds the connection object that aids in establishing connection with the specific ALM system. Please refer the *BaseAssetsManage.connect* method for more info.

BaseFields.project_info

This property holds information about a specific project from the list of selected projects, in the form of a Dict. The selected project will be one from the list of projects held in the *projects* property.

BaseFields.query_params

The input value which are selected by customer.

It returns Dict which contain information about APIs

Methods

BaseFields.fetch_fields() → List[Dict]

This method must be overridden and by the plugins to fetch information on each of the fields for the selected asset type in the respective ALM system and return in the form of a List of Dict. Failure to override this method by a plugin will result in an exception being raised.

BaseFields.fetch_additional_plugin_info() → Dict

This method maybe overridden and used by the plugins to store additional configuration that maybe needed during sync. If overridden, the method is expected to return a Dict, like a key-value pair. The base method returns an empty Dict.

BaseFields.skip_common_fields() → List

This method maybe overridden by the plugins to indicate fields that are not to be checked for commonality across selected projects when multiple projects are selected. If implemented this method must return a List of display names of the fields that should not be checked for commonality across projects. The base method returns an empty List meaning no fields will be ignored from the check for commonality.

1.2.4 Field — Field API

The Field class must be implemented by the respective plugins deriving from the BaseField class. This handles individual fields and processes them according to their respective types and formats as recognized by AgilitySync.

BaseField Class

class AgilitySync.mapping.BaseField(**args*, ***kwargs*)

This is the base class which must be inherited by the plugins as Field class to implement the methods that deal with various aspects of a particular field from the list of various fields for a particular asset/workitem type of the particular plugin instance.

Properties

BaseField.name

This property holds the name of the field as an str.

BaseField.display_name

This property holds the display name of the field as it is seen in the UI of the respective instance. The value is in the form of str.

BaseField.instance_obj

This property holds the connection object to the particular plugin instance. Please refer to *BaseAssetsManage.connect* for more info.

BaseField.payload_pattern

This property holds the pattern in which the field will be present in the payload received via webhook events from the respective instance of the plugin upon activities like workitem creation, update, comment etc. This will be used to extract the value for that particular field from the received payload. The response is expected as a Dict, with individual keys for create and update denoting the payload patterns for the create and update events respectively. Please see below for a sample response.

Sample Response:

```
{
    "create": Payload pattern of create event field,

    "update": Payload pattern of update event field
}
```

BaseField.reset_value

This property holds the value for the field which can be used to reset the value of this field. The response can be of any type as it depends on the type of the field.

BaseField.is_reset_value

This property holds the value of type bool which denotes if the extracted value for the field from the payload is the value used to reset this field's value.

BaseField.is_blank_value

This property holds a value of type bool which identifies if the extracted value for the field from the payload is blank or an equivalent to that with respect to the particular ALM system.

BaseField.blank_value

This property holds a value of any type depending on the type of the field, which indicates the value that is considered as blank value for this particular field by the respective plugin instance.

BaseField.is_readonly

This property holds a bool value denoting if the particular field is read-only or not. If the value held is True then it means the field is read-only and otherwise not.

BaseField.is_disabled

This property holds a bool value denoting if the field is disabled or not. A value of True means that the field is disabled and not otherwise.

BaseField.is_multivalue

This property holds a bool value representing if the field accepts multiple values or not. A value of True means it does and otherwise not.

BaseField.is_required

This property holds a bool value denoting if the field is a mandatory field or not. A value of True means the field is mandatory and otherwise not.

BaseField.has_default_value

This property holds a bool value which if True means that the field has a default value and otherwise not.

BaseField.is_custom

This property holds a bool value which if True means that the field is a custom/user defined field. If not, it means that the field is a part of the base system itself.

BaseField.workflow

This property holds in the form of a Dict any workflow that the field follows. If nothing of that sort is applicable to this field then the value will be None.

BaseField.fieldtype_info

This property holds information pertaining to the type of the particular field as a Dict.

BaseField.fields_obj

This property holds the Fields class of the particular plugin. This will be the class that particular plugin creates by inheriting from the *BaseFields* class.

BaseField.field_attr

This property holds the various attributes of the field in the form of a Dict.

BaseField.supported_additional_mapping

This property holds the list of entries for the additional mapping section. The response is expected as a List.

Methods

BaseField.fetch_name() → str

This method must be overridden by the plugin to return the name of the field as a str. Failure to override this method by the plugin results in an exception being raised.

BaseField.fetch_display_name() → str

This method must be overridden by the plugin to return the display name of the field, as it would be shown in the UI of the respective ALM system. The response must be a str. Failure to override this method by the plugin results in an exception being raised.

BaseField.is_required_field() → bool

This method must be overridden by the plugin to return a bool which should be True if the field is a mandatory field and False otherwise. Failure to override this field results in an exception being raised.

BaseField.is_multivalue_field() → bool

This method must be overridden to return a bool value indicating whether the field is multi values capable or not. True indicates that it is multi-value capable and False indicates otherwise. Failure to override this method by the plugin results in an exception being raised.

BaseField.is_disabled_field() → bool

This method must be overridden by the plugin to return a bool denoting whether this field is disabled. True indicates that the field is disabled and False indicates otherwise. Failure to override this method by the plugin results in an exception being raised.

BaseField.is_custom_field() → bool

This method must be overridden by the plugin to indicate whether this is a custom/user defined field or a base system field, by returning a bool which if True denotes that it is a custom field and otherwise not. Failure to override this method by the plugin results in an exception being raised.

BaseField.is_readonly_field() → bool

This method must be overridden by the plugin to return a bool indicating if this field is a read-only field. A value of True means that it is a read-only field and otherwise not. Failure to override this method by the plugin results in an exception being raised.

BaseField.is_support_default_value() → bool

This method may be overridden by the plugin to indicate if this field has a default or not by returning a bool which if True indicates that the field has a default value and otherwise not. This base method returns a default value of False.

BaseField.fetch_plugin_attr() → Dict

This method maybe overridden by the plugin to return as Dict any additional attributes that may be associated with this field. This base method returns an empty Dict.

BaseField.fetch_fieldtype_info() → Dict

This method must be overridden by the plugin to return the field's type info in the form of a Dict. Please see below for sample responses for different field types. Failure to override this method by the plugin results in an exception being raised.

Sample Response for list/Team (Dropdown fields):

```
{
    "type": "Type of the field. Refer '.FieldType' for support. Refer FieldType␣
↪Class
    "display_icon": "The icon which plugin wants to show in UI. Refer␣
↪FieldDisplayIcon"
                    "for supported icons"
    "values": {
        "id": "id of the value",
        "value": "This value would be used during sync",
        "display_value": "This value would be used to display the value in UI"
    },
    "value_type": FieldType.TEXT
}
```

Sample Response for other fields:

```
{
    "type": "Type of the field. Refer '.FieldType' for support. Refer FieldType␣
↪Class
    "display_icon": "The icon which plugin wants to show in UI. Refer␣
↪FieldDisplayIcon"
                    "for supported icons"
}
```

BaseField.fetch_supported_additional_mapping() → List

The plugin may override this method to return what is supported in the additional mappings section for this field. The response is expected as a List and this base method returns an empty List.

BaseField.fetch_workflow() → Dict

This method maybe overridden by the plugin to return the workflow information pertaining to this field in the form of a Dict. This base method returns an empty Dict. Please see below for a sample response.

Sample Response:

```
{
    "title": "Workflow title",
    "columns": [
        {
            "name": "name"
        }
    ],
    "workflow": [
        {
            "name": "name
        }
    ]
}
```

BaseField.fetch_fieldtype_expression_filename() → str

This method may be overridden by the plugin to return the filename to check for the type expression of the field. This method can return the name of the file in which to look for instead of the default value of "field-type_expression.json" which this base method returns.

1.2.5 WebHook — WebHook API

The WebHook class must be implemented by the respective plugins deriving from the BaseWebHook class. This handles individual webhook creation at the respective ALM system's instance while the BaseWebHook takes care of creating the receiver in the Agility Sync for receiving the webhooks.

BaseWebHook Class

class AgilitySync.mapping.BaseWebHook(**args*, ***kwargs*)

Properties

BaseWebHook.instance_obj

This property holds the connection object to the plugin instance and maybe used to perform operations on the instance, as type Any.

BaseWebHook.asset_name

This property holds the asset's name as Any.

BaseWebHook.instance_details

This property holds as a dict the details of the plugin instance.

BaseWebHook.project_ids

This property holds the list of project ids.

Methods

BaseWebHook.create_webhook(*webhook_name*, *webhook_url*, *webhook_description*, *project_id=None*) → None

This method must be overridden by the plugin to create a webhook on the ALM system's instance so it would send create/update/delete events for workitems happening on its' side. Failure by the plugin to override this method results in an exception being raised.

Parameters

- webhook_name (*str*) – Name of the webhook
- webhook_url (*str*) –Agility Syncwebhook URL to be added
- webhook_description (*str*) – Description of the webhook
- project_id (*Union[str|int]*) – Project level webhook

Returns

- None

BaseWebHook.update_webhook(*webhook_name*, *project_id=None*) → None

This method must be overridden by the plugin to update a webhook that was earlier created on the ALM system's instance by AgilitySync. Failure by the plugin to override this method results in an exception being raised.

Parameters

- webhook_name (*str*) – Name of the webhook
- webhook_url (*str*) –Agility Syncwebhook URL to be added

Returns

- None

1.3 Sync

1.3.1 Introduction

Sync is an Agility Sync Python package which contains classes and methods that can be used as is or overridden by the plugin as necessary to perform syncs between the mapped ALM systems.

The sync module contains the functionality that starts from processing an event that is registered with the Agility Sync via a webhook from a mapped ALM system. It figures out the specific mapping that is relevant to this event. If none is found, then the event is ignored. Upon finding the relevant mapping it initiates the processing of the inbound event, extraction of values for the fields that are mapped, normalizing the fields values to Agility Sync understandable format. It then initiates the outbound processing where the normalized data is taken and transformed into the format that the outbound system recognizes and then the sync attempt is made. The event is marked with the respective result of whether this sync was a success or not. In case of failure for some reason it provides for the ability to retry too, for any number of times. Another option available for failed events is the ability to cancel the event so further events between the 2 ALM systems for that particular mapping can be taken up for processing.

Each plugin must create sync.py file.

The plugin can access this package by importing AgilitySync.sync.

1.3.2 Payload — Payload API

Plugins must implement Payload class inheriting from the BasePayload class.

BasePayload Class

class AgilitySync.sync.BasePayload(**args*, ***kwargs*)

Properties

BasePayload.payload

This property holds the payload as it is received from the ALM system via webhooks. It is in the form of a Dict.

BasePayload.mapping_ids

This property holds a List of all the ids of the mappings created for this instance. The mapping ids will be of type bson.objectid.ObjectId.

BasePayload.instance_id

This property holds the instance ID of the plugin instance as a str.

BasePayload.db

This property holds the connection object that gives access to the DB.

Methods

BasePayload.fetch_events() → List[Dict]

This method maybe overridden by the plugin to identify events from the payload. The response is expected as a List of Dict. This base method returns the received payload inside a List.

BasePayload.fetch_project(*event*) → Union[int, str]

This method must be overridden by the plugin to take an event as argument to fetch and return the project ID from it. This project ID will be used in identifying the respective mapping during the sync. The returned value maybe of type int or str depending on the ALM system and its webhook payload. Failure by the plugin to override this method will result in an exception being raised.

Args: event (Dict): Event doc

Returns: project id: Union[int, str]

BasePayload.fetch_asset(*event*) → Union[str, int]

This method must be overridden by the plugin to take an event as argument to fetch and return the asset/workitem from it as either a str or an int depending on the respective ALM system and its webhook payload. This return value will be used to identify the corresponding mapping during the sync.

Args: event (Dict): Event doc

Returns: asset: Union[str, int]

BasePayload.fetch_query(*event*, *project*, *asset*) → List[Dict]

This method returns the query that will be used to identify the particular mapping which should be used to process the payload and perform the sync as there can be multiple such mappings for a particular instance. The response is in the form of a List of Dict.

In case the plugin intends to override the default mechanism used to identify the mapping it may override this method to suit its needs. A sample return of this base method is given below for reference and to help with formatting the response in case the plugin overrides this method.

Args: event (Dict): _description_ project_id (Union[str|int]): Project ID asset_type (Union[str|int]): Asset

Returns: List[Dict]

This method should return the below query.

Sample Response:

```
[
    {
        "$match": {
            "_id": {"$in": self.mapping_ids},
            "active": True,
            "$or": [
                {
                    "forward-direction.inbound.plugin_name": "teamforgeplugin",
                    "forward-direction.system_mapping.inbound_plugin_config.project_
→info.project": project,
                    "forward-direction.system_mapping.inbound_plugin_config.asset_
→info.asset": asset,
                    "direction_type": {"$in": ["forward-direction", "bi-direction"]}
→,
                    "forward-direction.inbound.instance_id": self.instance_id
                },
```

(Continues on next page)

(Continued from previous page)

```
                {
                    "backward-direction.inbound.plugin_name": "teamforgeplugin",
                    "backward-direction.system_mapping.inbound_plugin_config.
→project_info.project": project,
                    "backward-direction.system_mapping.inbound_plugin_config.asset_
→info.asset": asset,
                    "direction_type": {"$in": ["backward-direction", "bi-direction
→"]},
                    "backward-direction.inbound.instance_id":    self.instance_id
                },
            ]
        },
    },
    {
        "$project": {
            "name": 1,
            "forward-direction": {
                "inbound": 1, "outbound": 1, "additional_mapping": 1, "field_mapping
→": 1,
                "system_mapping": {
                    "$filter": {
                        "input": "$forward-direction.system_mapping",
                        "as": "system_mapping",
                        "cond": {
                            "$and": [
                                {"$eq": [
                                    "$$system_mapping.inbound_plugin_config.project_
→info.project",
                                    project]},
                                {"$eq": ["$forward-direction.inbound.instance_id",␣
→self.instance_id]},
                                {"$eq": ["$$system_mapping.inbound_plugin_config.
→asset_info.asset", asset]},
                                {"$eq": ["$forward-direction.inbound.plugin_name",␣
→plugin_name]}
                            ]
                        }
                    }
                }
            },
            "backward-direction": {
                "inbound": 1, "outbound": 1, "additional_mapping": 1, "field_mapping
→": 1,
                "system_mapping": {
                    "$filter": {
                        "input": "$backward-direction.system_mapping",
                        "as": "system_mapping",
                        "cond": {
                            "$and": [
                                {"$eq": [
                                    "$$system_mapping.inbound_plugin_config.project_
→info.project",
```

(Continues on next page)

(Continued from previous page)

```
                                    project]},
                                {"$eq": ["$backward-direction.inbound.instance_id",␣
→self.instance_id]},
                                {"$eq": ["$$system_mapping.inbound_plugin_config.
→asset_info.asset", asset]},
                                {"$eq": ["$backward-direction.inbound.plugin_name",␣
→plugin_name]}
                            ]
                        }
                    }
                }
            }
        }
    }
]
```

BasePayload.is_cyclic_event(*event*, *sync_user*) → bool

The plugin must override this method to return a bool indicating whether this event is cyclic. A value of True means that the event is cyclic and will be skipped hence. Otherwise, it will be assumed that the event is not cyclic and Agility Sync will move forward with processing it. Basically the overridden method is expected to check if this particular event is one that was generated by an action performed by the user who's configured as the sync user for this particular plugin instance and return True if so and False otherwise so that cyclic events can be avoided. Failure by the plugin to override this method results in an exception being raised.

Args: event (_type_): _description_ sync_user (_type_): _description_

Returns: bool

BasePayload.fetch_system_mapping(*mapping_doc*, *_event*) → dict

This method identifies and returns the specific system mapping from a possible list of system mappings for the same project on the ALM system. In such a case the plugin may override this method to identify and return the specific system mapping from that list and return it as a Dict. This base method assumes that only one matching system mapping has been identified and uses the first element in the system mapping List, which is the most likely scenario unless the plugin allows the same project to be selected multiple times in the mapping. Such a case is when the plugin is expected to override this method and return the correct mapping.

Args: mapping_doc (_type_): _description_ _event (_type_): _description_

Returns: dict: _description_

1.3.3 Event — Event API

Plugins must implement Event class inheriting from the BaseEvent class.

BaseEvent Class

class AgilitySync.sync.BaseEvent(**args*, ***kwargs*)

This class must be inherited by the respective plugins as Event to add methods to handle the event once it has been identified by the Payload class. This base class contains generic methods to handle the event like writing the event to the DB etc.

Properties

BaseEvent.event_type

This property holds the type of this particular event, which will be one of those listed in *AgilitySync.constants.EventTypes*.

BaseEvent.event_id

This property holds the unique ID of this particular event in str format. It will be the ID generated upon the event being inserted to the respective collection in the DB.

BaseEvent.datamap

This property holds the datamap that matches the payload and contains only the matching content and the direction too, in the form of a Dict.

BaseEvent.event

This property holds the event in the form of a Dict.

BaseEvent.workitem_id

This property holds the ID of the workitem for which this event was generated as either an str or an int depending on how the particular ALM system maintains it.

Methods

BaseEvent.fetch_workitem_id() → Union[str, int]

This method must be overridden by the plugin to return the workitem's ID for which this event was generated, in the form of str or int based on how the respective ALM system maintains it. Failure by the plugin to override this method results in an exception being raised.

BaseEvent.fetch_workitem_display_id() → str

This method must be overridden by the plugin to return the workitem's display ID as it would be displayed in the UI of the respective ALM system, in the form of an str. Failure by the plugin to override this method results in an exception being raised.

BaseEvent.fetch_workitem_url() → str

This method must be overridden by the plugin to return the workitem's URL as an str. Failure by the plugin to override this method will result in an exception being raised.

BaseEvent.fetch_event_type() → *AgilitySync.constants.EventTypes*

The plugin must override this method to return the event type of this particular event. It must be of one of the types defined in *AgilitySync.constants.EventTypes*. Failure by the plugin to override this method will result in an exception being raised.

BaseEvent.fetch_timestamp() → datetime.datetime

This method must be overridden by the plugin to return as datetime the timestamp at which the action this triggered this event happened at the source ALM system. The response must be in **UTC** format. Failure to override this method by the plugin will result in an exception being raised.

BaseEvent.fetch_moved_project_id() → Any

The plugin may override this method to identify and return the new project ID in case a workitem has been moved from one project to another. This base method returns an empty str. However, the return type from the overridden method maybe of type Any.

BaseEvent.fetch_revision() → Any

The plugin may override this method to return something that uniquely identifies this event. This helps in avoiding processing the same event twice in case multiple webhook events are received by AS for the same action at the ALM system's side. This base method returns an empty str. However, the return type from the overridden method maybe Any.

BaseEvent.fetch_schedule_time() → datetime.datetime

This method returns the present UTC time as events are scheduled for processing as and when they are received and formatted accordingly. If a plugin wishes to introduce a slight delay in the handling of a particular type of event likely to allow another event from the same instance to be processed first, then it may do so by adding a small delay to the current UTC time and returning it. This would come in handy in cases where a particular system always sends 2 particular events out of order, and you want to process 2 events you received in the reverse order. The response from this method must be of datetime in **UTC** format.

1.3.4 Inbound — Inbound API

Plugins must implement Inbound class inheriting from the BaseInbound class.

BaseInbound Class

class AgilitySync.sync.BaseInbound(*args, **kwargs)

This class must be inherited by the plugins as Inbound class, and it should contain the methods to process the normalization of the inbound event for being synced later.

Properties

BaseInbound.instance_id

This property holds the unique ID of the plugin instance as an str.

BaseInbound.instance_object

This property holds the connection object to the instance of the plugin. It is the return value of the *BaseInbound.connect* method and is of type Any.

BaseInbound.event_timestamp

This property holds the time stamp of the event being handled, as datetime in UTC format.

BaseInbound.instance_name

This property holds the name of the instance as configured in the plugins, as an str.

BaseInbound.system_type

This property holds the type of the plugin that this instance is of, as an str.

BaseInbound.event_info

This property holds the event under process, in the form of a Dict.

BaseInbound.xref_id

This property returns the unique ID that identifies the sync reference record created in the DB. This record holds information about the respective workitems from the 2 ALM systems in a mapping that are to be maintained in sync by AS. It will be an bson.objectid.ObjectId.

BaseInbound.assets_info

This property holds the project and asset configuration for the instance, as Dict.

BaseInbound.workitem_id

This property holds the workitem's ID as an str or int depending on how the ALM system manages it.

BaseInbound.workitem_display_id

This property holds the display ID of the workitem as it is seen in the UI of the respective ALM system instance. It's of type str.

BaseInbound.workitem_url

This property holds the workitem's URL as an str.

BaseInbound.project_info

This property holds information about the project on the instance for which the event is being processed. It's of type Dict.

BaseInbound.asset_info

This property holds information about the asset on the instance for which the event is being processed. It's of type Dict.

BaseInbound.event_type

This property holds the type of the event which is being processed, as an str.

BaseInbound.received_by

BaseInbound.plugin_name

This property holds the name of the plugin whose instance this event has come from, as an str.

BaseInbound.direction_type

This property holds the direction type from the mapping that this event must be processed in, as an str.

BaseInbound.field_mapping

This property holds information about all the fields that are mapped in this mapping, as a List of Dict.

BaseInbound.db

BaseInbound.normalized_fields

This property holds the fields with their values in normalized format as understood by AS. It's a List of Dict.

BaseInbound.event

This property holds the event as it is extracted from the payload and is being processed, in the form of a Dict.

BaseInbound.status

This property holds the status of the sync attempted to the ALM system mapped on the other side.

BaseInbound.event_id

This property holds the unique ID of the event that is being processed. It is of type bson.objectid.ObjectId.

BaseInbound.comment

This property holds the normalized content from the comments section of the workitem, retrieved from the event. It's of type str.

BaseInbound.absoulte_workitem_url

This property holds the absolute URL to the workitem on the ALM system's instance. It's of type str.

BaseInbound.parent_id

This property holds the unique ID of the workitem on the instance which the workitem pertaining to the event under process is related to. The type is either str or int depending on how the ALM system handles it.

BaseInbound.additional_mapping

This property holds information on the supported additional mapping for this particular mapping, in the form of a Dict.

BaseInbound.old_parent_id

This property holds the ID of the workitem on the instance to which the workitem pertaining to the event under process now was related to earlier and has changed now. It could be of type str or int depending on how the respective ALM system manages it.

BaseInbound.datamap

This property holds the datamap which holds the mapping for how this event has to be processed and synced. It also holds the selective matched parameters based on which project the event is from etc. It will be of the form Dict.

Methods

BaseInbound.fetch_event_category() → List[*AgilitySync.constants.EventCategory*]

This method must be overridden by the plugin to return a List of *AgilitySync.constants.EventCategory* that is part of the event under process. Please refer *AgilitySync.constants.EventCategory* for more information. Failure by the plugin to override this method results in an exception being raised.

Returns: List[EventCategory]: _description_

BaseInbound.connect() → Any

This method must be overridden by the plugin to return a connection object to the respective instance. This will be stored and used for performing needed operations on the instance. It's of type Any. It can be accessed using the property *BaseInbound.instance_object*. Failure to override this method by the plugin will result in an exception being raised.

BaseInbound.fetch_parent_id() → Tuple[Optional[Union[int, str]], Optional[Union[int, str]]]

This method must be overridden by the plugin in case relationship is supported and expected to be carried over by AS to the other ALM system too. This method must return the ID of the related workitem in the form of a Tuple with the related workitem's ID and earlier related workitem's id. Failure by the plugin to override this method if relationships are supported, results in an exception being raised.

BaseInbound.fetch_attachments_metadata() → List[Dict]

The plugin must override this method to return as a List of Dict the attachments related metadata for each attachment, in case attachments are supported by this plugin. Failure by the plugin to override this method in case attachments are supported will result in an exception being raised. A sample response is given below for reference.

Sample Response:

```
[
   {
        "type"(str): "ADDED or REMOVED" ("ADDED" - For newly added attachment,
                                          "REMOVED" - For removed attachment)

        "filename"(str): "filename of the attachment",

        "url"(str): The download attachment url,

        "id"(Union[str,int]): "Unique Id of the attachment",

        "content_type"(str): Type of the content.

        "headers"(Dict, optional): Plugin can set this value if any additional␣
→headers which needs to
                                    passed while downloading attachment.

        "basic_auth"(Tuple, optional): Plugin can set this value if plugin uses␣
→basic authentication.
                                   First value contain ''username'' and second␣
→value contain
                                   ''password''
   }
]
```

BaseInbound.fetch_comment() → str

This method must be overridden by the plugin to fetch the value for comments from the event, if comments are supported by the plugin. The expected return type is str. Failure to override this method by the plugin results in an exception being raised.

BaseInbound.fetch_dependencies()

The plugin must override this method to fetch and return any dependencies identified from the event, if the plugin supports dependencies. Failure by the plugin to override this method results in an exception being raised.

BaseInbound.mapped_fieldnames() → List[str]

This method returns the list of mapped fields names as a List of str.

BaseInbound.create_remote_link(*display_id, url, id_field: Optional[dict] = None, url_field: Optional[dict] = None*) → None

The plugin must override this method in case sync reference is supported for this plugin. This method would be called after the sync to the ALM system on the other side is successful. It is expected to take the display ID and URL of the workitem created on the other ALM system and optional parameters of id_field and url_field which denote the fields to which the respective data must be updated to on the source ALM system. This base method does nothing and so any action will happen only if the method is overridden by the plugin.

Args: display_id (str): synced workitem display ID URL (str): synced workitem URL id_field (dict, optional): Synced workitem display ID should be set to this field. Defaults to None. url_field (dict, optional): Synced workitem URL should be set to this field. Defaults to None.

BaseInbound.normalize_create_fields(*inbound_field_attr: dict*) → Tuple[bool, Any]

The plugin may override this method if it intends to deviate from the default behavior of using the payload pattern to fetch the value for fields from the event, as it is received from the ALM instance. It takes as input a Dict containing information on the fields for which normalized values must be fetched. This method is specific to create type events only. The return value expected is a Tuple of two values, a bool which if True indicates

that this field is present in the received event, and the second being the value of the field as retrieved from the event and is of type Any.

Args: inbound_field_attr (Dict): Information of the field.

Returns: Tuple[bool, Any]

BaseInbound.normalize_update_fields(*inbound_field_attr*)

The plugin may override this method if it intends to deviate from the default behavior of using the payload pattern to fetch the value for fields from the event, as it is received from the ALM instance. It takes as input a Dict containing information on the fields for which normalized values must be fetched. This method is specific to update type events only. The return value expected is a Tuple of two values, a bool which if True indicates that this field is present in the received event, and the second being the value of the field as retrieved from the event and is of type Any. By default, this value would be fetched by using payload_pattern expression value.

Args: inbound_field_attr (Dict): Information of the field.

Returns: Tuple[bool, Any]

BaseInbound.normalize_texttype_fieldvalue(*value: Any*, *field_attr: Dict*) → str

The plugin may override this method if further normalization of texttype fields is needed. The arguments to be taken are the value of the text field and the fields information Dict. It then must return the value as str.

Args: value (Any): Received value from event. field_attr (Dict): field_attr

Returns: str

BaseInbound.normalize_urltype_fieldvalue(*value: Any*, *field_attr: Dict*) → str

The plugin may override this method if further normalization of urltype fields is needed. The arguments to be taken are the value of the URL field and the fields information Dict. It then must return the value as str.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: str

BaseInbound.normalize_textareatype_fieldvalue(*value: Any*, *field_attr: Dict*) → str

The plugin may override this method if further normalization of textareatype fields is needed. The arguments to be taken are the value of the textarea field and the fields information Dict. It then must return the value as str.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: str

BaseInbound.normalize_htmltype_fieldvalue(*value: Any*, *field_attr: Dict*) → str

The plugin may override this method if further normalization of htmltype fields is needed. The arguments to be taken are the value of the html field and the fields information Dict. It then must return the value as str.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: str

BaseInbound.normalize_numerictype_fieldvalue(*value: Any*, *field_attr: Dict*) → Union[int, float]

The plugin may override this method if further normalization of numerictype fields is needed. The arguments to be taken are the value of the numeric field and the fields information Dict. It then must return the value as int or float.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: Union[int, float]

BaseInbound.normalize_usertype_fieldvalue(*value: Any*, *field_attr: Dict*) → Dict[str, str]

The plugin may override this method if further normalization of usertype fields is needed. The arguments to be taken are the value of the user field and the fields information Dict. The expected return value is a Dict of key-value pair type with both being of type str.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: Dict[str, str]

A sample response is given below for reference.

Sample Response:

```
{
    "username"(str): "name of the username"

    "email"(str): "Email address of the username"
}
```

BaseInbound.normalize_timetype_fieldvalue(*value: Any*, *field_attr: Dict*) → str

The plugin may override this method if further normalization of timetype fields is needed. The arguments to be taken are the value of the time field and the fields information Dict. It then must return the value as str of the format "HH:MM:SS".

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: str

BaseInbound.normalize_datetype_fieldvalue(*value: Any*, *field_attr: Dict*) → datetime.datetime

The plugin may override this method if further normalization of datetimetype fields is needed. The arguments to be taken are the value of the datetime field and the fields information Dict. It then must return the value as datetime.datetime.isoformat.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: datetime

BaseInbound.normalize_datetimetype_fieldvalue(*value: Any*, *field_attr: Dict*) → datetime.datetime

The plugin may override this method if further normalization of datetype fields is needed. The arguments to be taken are the value of the date field and the fields information Dict. It then must return the value as datetime.datetime.isoformat.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: datetime

BaseInbound.normalize_listtype_fieldvalue(*value: Any*, *field_attr: Dict*) → Union[str, int]

The plugin may override this method if further normalization of listtype fields is needed. The arguments to be taken are the value of the list field and the fields information Dict. It then must return the value as str or int.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: Union[str, int]

BaseInbound.normalize_teamtype_fieldvalue(*value: Any*, *field_attr: Dict*) → Union[str, int]

The plugin may override this method if further normalization of teamtype fields is needed. The arguments to be taken are the value of the team field and the fields information Dict. It then must return the value as str or int.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: Union[str, int]

BaseInbound.normalize_booltype_fieldvalue(*value: Any*, *field_attr: Dict*) → bool

The plugin may override this method if further normalization of booleantype fields is needed. The arguments to be taken are the value of the Boolean field and the fields information Dict. It then must return the value as bool.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: bool

BaseInbound.normalize_texttype_multivalue_field(*value: Any*, *field_attr: Dict*) → List[Dict[str, str]]

The plugin may override this method if further normalization of text multivalue fields are needed. The arguments to be taken are the value of the text multivalue field and the fields information Dict. The return value must be of type List of Dict in the form of key-value pairs where both are of type str.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: List[Dict[str, str]]

Please see below for a sample response for reference.

Sample Response:

```
[
   {
       "act"(str): "add" or "remove"

       "field_value"(str):  value
    }
]
```

BaseInbound.normalize_urltype_multivalue_field(*value: Any*, *field_attr: Dict*) → List[Dict[str, str]]

The plugin may override this method if further normalization of URL multivalue fields are needed. The arguments to be taken are the value of the URL multivalue field and the fields information Dict. The return value must be of type List of Dict in the form of key-value pairs where both are of type str.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: List[Dict[str, str]]

Please see below for a sample response for reference.

Sample Response:

```
[
   {
       "act"(str): "add" or "remove"

       "field_value"(str):  value
    }
]
```

BaseInbound.normalize_usertype_multivalue_field(*value: Any*, *field_attr: Dict*) → List[Dict[str, str]]

The plugin may override this method if further normalization of user multivalue fields are needed. The arguments to be taken are the value of the user multivalue field and the fields information Dict. The return value must be of type List of Dict in the form of key-value pairs where both are of type str.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: List[Dict[str, str]]

Please see below for a sample response for reference.

Sample Response:

```
[
    {
        "username"(str): "name of the username"

        "email"(str): "Email address of the username"
    }
]
```

BaseInbound.normalize_listtype_multivalue_field(*value: Any*, *field_attr: Dict*) → List[Union[str, int]]

The plugin may override this method if further normalization of list multivalue fields are needed. The arguments to be taken are the value of the list multivalue field and the fields information Dict. The return value must be of type List of Dict in the form of key-value pairs of types str and int respectively.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: List[Dict[str, int]]

Please see below for a sample response for reference.

Sample Response:

```
[
    {
        "act"(str): "add" or "remove"

        "field_value"(str): "field value"
    }
]
```

BaseInbound.normalize_teamtype_multivalue_field(*value: Any*, *field_attr: Dict*) → List[Union[str, int]]

The plugin may override this method if further normalization of team multivalue fields are needed. The arguments to be taken are the value of the team multivalue field and the fields information Dict. The return value must be of type List of Dict in the form of key-value pairs of types str and int respectively.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: List[Dict[str, int]]

Please see below for a sample response for reference.

Sample Response:

```
[
    {
        "act"(str): "add" or "remove"

        "field_value"(str): "field value"
    }
]
```

BaseInbound.normalize_htmltype_multivalue_field(*value: Any*, *field_attr: Dict*) → List[Dict]

The plugin may override this method if further normalization of html multivalue fields are needed. The arguments to be taken are the value of the html multivalue field and the fields information Dict. The return value must be a List of type Dict. Plugin may override this function to further normalize the value of html field type.

Args: value (Any): Received value from event. field_attr (Dict): field attr

Returns: List[Dict]

Please see below for a sample response for reference.

Sample Response:

```
[
    {
        "act"(str): "add" or "remove"

        "field_value"(str): "field value"
    }
]
```

BaseInbound.migrate_create() → List[Dict]

This method must be overridden by the plugin to handle creation of a workitem whose update event has been received but was no sync reference information is available with AS probably due to the creation happening before the mapping being created or the creation webhook event not being received. This method must construct an event as a create event from the details present in the received update event and return it in the form of a List

of Dict. Failure to override this method by the plugin results in an exception being raised.

BaseInbound.migrate_update() → List[Dict]

This method must be overridden by the plugin to handle an update after some period of being idle with respect to this particular mapping like when the mapping was made inactive for some duration. In such a case there may be updates that happened during the time of the mapping being inactive. This method must collect all updates and create a migrate update event with all the updates in the form of a List of Dict. Failure by the plugin to override this method results in an exception being raised.

Class Methods

classmethod BaseInbound.inbound_class(*instance_details*) → AgilitySync.sync.inbound.T

This method maybe overridden by the plugin if it intends to use some class other than its Inbound class for creating the inbound class object. This method must return that class that must be used.

Args: instance_details (Dict): instance details

1.3.5 Outbound — Outbound API

Plugins must implement Outbound class inheriting from the BaseOutbound class.

BaseOutbound Class

class AgilitySync.sync.BaseOutbound(**args*, ***kwargs*)

This class must be inherited by the plugins as OutBound and it should contain the methods to further process the normalized event as per the needs of the outbound ALM system for the sync to be successful.

Properties

BaseOutbound.system_type

This property holds the type of the plugin that this instance is of, as an str.

BaseOutbound.direction_type

This property holds the direction type from the mapping that this event must be processed in, as an str.

BaseOutbound.assets_info

This property holds the project and asset configuration for the instance, as Dict.

BaseOutbound.additional_mapping

This property holds information on the supported additional mapping for this particular mapping, in the form of a Dict.

BaseOutbound.inbound

This property holds the instance of the inbound object that corresponds to this outbound instance. It is of type *BaseInbound*.

BaseOutbound.plugin_name

This property holds the name of the plugin whose instance this event has come from, as an str.

BaseOutbound.instance_id

This property holds the unique ID of the plugin instance as an str.

BaseOutbound.instance_name

This property holds the name of the instance as configured in the plugins, as an str.

BaseOutbound.instance_object

This property holds the connection object to the instance of the plugin. It's the return value of the *BaseInbound.connect* method and is of type Any.

BaseOutbound.project_info

This property holds information about the project on the instance for which the event is being processed. It's of type Dict.

BaseOutbound.asset_info

This property holds information about the asset on the instance for which the event is being processed. It's of type Dict.

BaseOutbound.workitem_id

This property holds the workitem's ID as an str or int depending on how the ALM system manages it.

BaseOutbound.workitem_display_id

This property holds the display ID of the workitem as it is seen in the UI of the respective ALM system instance. It's of type str.

BaseOutbound.outbound_id

This property holds the unique ID of the outbound instance as in the DB. It's stored as *bson.ObjectId*.

Methods

BaseOutbound.create(*sync_fields*) → Dict

This method must be overridden by the plugin to receive the fields with respective values as a Dict and handle creation of the workitem on the outbound ALM system. Upon successful creation of the workitem the method is expected to return a Dict with contents as detailed below. Failure by the plugin to override this method results in an exception being raised.

Sample Response:

```
{
    "id": "Newly created workitem id. This ID should be unique for your instance",
    "display_id": "workitem display id",
    "relative_url": "Relative URL of the created workitem",
    "absolute_url": "Absolute URL of the created workitem"
}
```

BaseOutbound.delete() → None

This method must be overridden by the plugin to handle deletion of the workitem in the outbound ALM system as a result of its corresponding workitem on the other ALM system being deleted. Failure by the plugin to override this method results in an exception being raised.

BaseOutbound.update(*sync_fields: dict*) → Tuple[Dict, datetime.datetime]

This method must be overridden by the plugin to receive the fields with respective values as a Dict and handle creation of the workitem on the outbound ALM system. Upon successful creation of the workitem the method is expected to return a Dict with contents as detailed below. Failure by the plugin to override this method results in an exception being raised.

BaseOutbound.connect() → Any

This method must be overridden by the plugin to return a connection object to the respective instance. This will be stored and used for performing needed operations on the instance. It's of type Any. It can be accessed using the property *BaseOutbound.instance_object*. Failure to override this method by the plugin will result in an exception being raised.

BaseOutbound.comment_create(*comment*) → None

This method may be overridden by the plugin to handle creation of comments on the outbound ALM System's instance, if comments are supported for this plugin type.

Args: comment (str): Received comment

BaseOutbound.dependency_create(*dependant_workitem_obj*, *source_workitem_obj*) → None

This method must be overridden by the plugin to create dependencies as supported by the outbound ALM system's instance. It takes the dependent and source workitems as arguments. Failure by the plugin to override this method results in an exception being raised, if dependencies are supported.

Args: dependant_workitem_obj (_type_): _description_ source_workitem_obj (_type_): _description_

BaseOutbound.dependency_update(*source_workitem_obj*, *dependent_parent_id*) → None

This method must be overridden by the plugin to update dependencies as supported by the outbound ALM system's instance. It takes the dependent and source workitems as arguments. Failure by the plugin to override this method results in an exception being raised, if dependencies are supported.

Args: dependant_workitem_obj (_type_): _description_ source_workitem_obj (_type_): _description_

BaseOutbound.dependency_delete(*dependant_obj*, *source_obj*, *dependency_id*) → None

This method must be overridden by the plugin to update dependencies as supported by the outbound ALM

system's instance. It takes the dependent and source workitems as arguments. Failure by the plugin to override this method results in an exception being raised, if dependencies are supported.

Args: dependant_obj (_type_): _description_ source_obj (_type_): _description_ dependency_id (_type_): _description_

Raises: exceptions.OutboundError: _description_

BaseOutbound.transform_old_parent_id() → Optional[Any]

This method transforms the old parent ID and is used in case the parent ID changes. It returns Any or None.

Returns: Union[Any, None]: _description_

BaseOutbound.transform_parent_id() → Optional[Any]

This method transforms the parent id. and returns either Any or None.

BaseOutbound.get_workitem_info()

This method returns the workitem information from the sync reference with respect to the outbound ALM system.

BaseOutbound.create_remote_link(*workitem_id*, *url*, *id_field: Optional[dict] = None*, *url_field: Optional[dict] = None*) → None

The plugin may override this method to help with creating remote links if they are supported for that particular type of plugin. This base function does nothing. Arguments to this method are as showcased below. The method is not expected to return anything.

Args: workitem_id (_type_): _description_ URL (_type_): _description_ id_field (dict, optional): _description_. Defaults to None. url_field (_type_, optional): _description_. Defaults to None.

BaseOutbound.transform_fields(*transform_field_objs*) → Dict

This method must be overridden by the plugin to transform all normalized fields, which it will receive as argument, to the format that is expected by the outbound ALM system's instance and returns it as a Dict. Failure by the plugin to override this method in an exception being raised.

Returns: Dict: _description_

BaseOutbound.transform_listtype_value(*value*, *field_obj*) → Any

This method maybe overridden by the plugin if it intends to do any additional transformations to values of listtype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Raises: exceptions.SkipFieldToSync: Plugin may raise this exception, if they want to skip field

Returns: Any: _description_

BaseOutbound.transform_listtype_multivalue(*value*, *field_obj*) → List[Dict]

This method maybe overridden by the plugin if it intends to do any additional transformations to values of listtype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Raises: exceptions.SkipFieldToSync: Plugin may raise this exception, if they want to skip field

Returns: List[Dict]: _description_

BaseOutbound.transform_texttype_value(*value*, *field_obj*) → str

This method maybe overridden by the plugin if it intends to do any additional transformations to values of texttype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Raises: exceptions.SkipFieldToSync: Plugin may raise this exception, if they want to skip field

Returns: _type_: _description_

BaseOutbound.transform_textareatype_value(*value*, *field_obj*) → str

This method maybe overridden by the plugin if it intends to do any additional transformations to values of textareatype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Raises: exceptions.SkipFieldToSync: Plugin may raise this exception, if they want to skip field

Returns: _type_: _description_

BaseOutbound.transform_numerictype_value(*value*, *field_obj*) → str

This method maybe overridden by the plugin if it intends to do any additional transformations to values of numerictype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Raises: exceptions.SkipFieldToSync: Plugin may raise this exception, if they want to skip field

Returns: _type_: _description_

BaseOutbound.transform_texttype_multivalue(*value*, *field_obj*) → List[Dict]

This method maybe overridden by the plugin if it intends to do any additional transformations to values of texttype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: List[Dict]: _description_

BaseOutbound.transform_htmltype_value(*value*, *field_obj*) → str

This method maybe overridden by the plugin if it intends to do any additional transformations to values of htmltype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: str: _description_

BaseOutbound.transform_htmltype_multivalue(*value*, *field_obj*) → List[str]

This method maybe overridden by the plugin if it intends to do any additional transformations to values of htmltype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: List[str]: _description_

BaseOutbound.transform_urltype_value(*value*, *field_obj*) → str

This method maybe overridden by the plugin if it intends to do any additional transformations to values of urltype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: List[str]: _description_

BaseOutbound.transform_urltype_multivalue(*value*, *field_obj*) → List[Dict]

This method maybe overridden by the plugin if it intends to do any additional transformations to values of urltype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: List[str]: _description_

BaseOutbound.transform_usertype_value(*value*, *field_obj*) → Dict

This method maybe overridden by the plugin if it intends to do any additional transformations to values of usertype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_usertype_multivalue(*value*, *field_obj*) → List[Dict]

This method maybe overridden by the plugin if it intends to do any additional transformations to values of usertype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_datetype_value(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of datetype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_datetype_multivalue(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of datetype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_datetimetype_value(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of datetimetype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_datetimetype_multivalue(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of datetimetype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_timetype_value(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of timetype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_timetype_multivalue(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of timetype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_teamtype_value(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of teamtype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_teamtype_multivalue(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of teamtype multivalue fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Returns: Dict: _description_

BaseOutbound.transform_booleantype_value(*value*, *field_obj*)

This method maybe overridden by the plugin if it intends to do any additional transformations to values of booleantype fields. The arguments are the value and the field object. This base method returns the value as it is.

Args: value (_type_): _description_ field_obj (_type_): _description_

Raises: exceptions.SkipFieldToSync: Plugin may raise this exception, if they want to skip field

Returns: _type_: _description_

1.3.6 Attachment — Attachment API

BaseAttachmentDownload Class

Plugins may implement AttachmentDownload class inheriting from the BaseAttachmentDownload class if they intend to change the way Agility Sync handles downloading attachments from the source ALM system's instance, in case the plugin supports attachments sync. This is however not necessary in most cases as the base class can handle it.

class AgilitySync.sync.BaseAttachmentDownload(**args*, ***kwargs*)

Properties

BaseAttachmentDownload.chunk_size

This property holds the chunk size which will determine the size of individual chunks of the attachment to be downloaded at a time.

BaseAttachmentDownload.basic_auth

This property holds the basic authentication.

BaseAttachmentDownload.headers

This property holds the required headers.

BaseAttachmentDownload.cookies

This property holds the required cookies.

BaseAttachmentDownload.type

This property holds the type of the attachment.

BaseAttachmentDownload.id

This property holds the ID of the attachment.

BaseAttachmentDownload.content_type

This property holds the content type of the attachment.

BaseAttachmentDownload.absolute_filename

This property holds the absolute filename of the attachment.

BaseAttachmentDownload.filename

This property holds the filename of the attachment.

BaseAttachmentDownload.url

This property holds the URL of the attachment.

Methods

BaseAttachmentDownload.download(*storage_obj*, *verity_tls*) → None

This method maybe overridden by the plugin if it has decided to inherit the BaseAttachmentDownload class as the AttachmentDowlnoad class. This method must contain the code to download the attachment from the source plugin instance. This base method handles the requirements itself and so in most cases, the plugin will not be required to override the class or this method.

BaseAttachmentUpload Class

Plugins must implement AttachmentUpload class inheriting from the BaseAttachmentUpload class to handle uploading an attachment to the target ALM system instance as part of syncing attachments if the plugin supports it.

class AgilitySync.sync.BaseAttachmentUpload(**args*, ***kwargs*)

Properties

BaseAttachmentUpload.instance_details

This property holds the details of the instance as a dict.

BaseAttachmentUpload.headers

This property holds the headers as a dict.

BaseAttachmentUpload.cookies

This property holds the cookies as a dict.

BaseAttachmentUpload.instance_object

This property holds a connection object to the plugin instance which may be used to perform operations on the instance.

BaseAttachmentUpload.content_type

This property holds the content type of the attachment as an str.

BaseAttachmentUpload.filename

This property holds the filename of the attachment as an str.

BaseAttachmentUpload.workitem_id

This property holds the ID of the workitem to which this attachment belongs to, as a str or int.

BaseAttachmentUpload.upload_url

This property holds the URL to upload the attachment to as an str.

BaseAttachmentUpload.outbound

This property holds the outbound object as a dict.

Methods

BaseAttachmentUpload.upload(*storage_obj*, *verify_tls: bool*) → None

This method maybe overridden by the plugin if it intends to change the functionality provided by this base method.

Parameters

- storage_obj (*BinaryIO*) – file object to store the attachment content.
- verify_tls (*bool*) – True/False

Returns

- None

BaseAttachmentUpload.upload_on_success(*response: dict*) → None

This is a callback method that the plugin may override if it intends to perform some action upon the attachment upload being successful. This base method does nothing and just returns.

BaseAttachmentUpload.upload_on_failed() → None

This is a callback method that the plugin may override if it intends to perform some action upon failure to upload the attachment. This base method does nothing and just returns.

BaseAttachmentUpload.remove(*attachment_id: Union[str, int]*, *verify_tls: bool*)

This method must be overridden by the plugin to handle attachment deletion. Failure by the plugin to override this method results in an exception being raised.

Parameters

- attachment_id (*Union[str, int]*) – Created attachment id
- verify_tls (*bool*) – True/False

Returns

- None

BaseAttachmentUpload.fetch_url() → str

This method must be overridden by the plugin to return the URL of the uploaded attachment. This method would be called upon when the attachment upload is successful. Failure by the plugin to override this method results in an exception being raised.

This method should return attachment URL.

BaseAttachmentUpload.fetch_upload_url() → str

This method must be overridden by the plugin to return the default upload URL for the attachment. Plugin must override this function to get the upload URL to upload by default. Failure by the plugin to override this method will result in an exception being raised.

The method must return location of the attachment upload URL to upload the attachment.

BaseAttachmentUpload.fetch_id() → Union[str, int]

This method must be overridden by the plugin to return the attachment's id. This method would be called upon after the attachment upload is successful. Failure by the plugin to override this method results in an exception being raised.

This method should return attachment unique id

BaseAttachmentUpload.fetch_header_info() → dict

The plugin may override this method if any additional headers are needed and return them in the form of a dict.

Returns

- Dict - header information

BaseAttachmentUpload.fetch_cookies() → Optional[dict]

The plugin may override this method to return cookies if needed, as a dict. This base method returns None.

Returns

- Dict/None - Either Dict or None

BaseAttachmentUpload.fetch_multipart_data(*storage_obj*) → Optional[dict]

The plugin may override this method to return parts of the data as dict.

This method must return either Dict or None

Parameters storage_obj (*BinaryIO*) – file object to store the attachment content.

1.4 Constants —Agility Sync Constants

class AgilitySync.constants.EventTypes

This class defines constants that identify different event types that are supported by AgilitySync.

CREATE = 'CREATE'

For workitem create event, we can use this constant

UPDATE = 'UPDATE'

For workitem update or attachment create, we can use this constant

DELETE = 'DELETE'

For workitem delete, we can use this constant

class AgilitySync.constants.EventCategory

This class defines constants that identify different event categories that are supported by AgilitySync.

ATTACHMENT = 'ATTACHMENT'

If the event contains attachments, this constant should be set.

WORKITEM = 'WORKITEM'

If the event contains workitem create or update, this constant should be set.

COMMENT = 'COMMENT'

If the event contains a comment, this constant should be set.

DEPENDENCY = 'DEPENDENCY'

If the event contains dependency, this constant should be set.

class AgilitySync.constants.SyncStatus

This class defines constants that hold the different possible statuses of a sync attempt.

SUCCESS = 'SUCCESS'

Event is successfully synced.

FAILED = 'FAILED'

Event is Failed.

PROCESSING = 'PROCESSING'

Event is processing.

QUEUED = 'QUEUED'

Event is queued.

SUBMITTED = 'SUBMITTED'

Event is submitted.

CANCELLED = 'CANCELLED'

Event is cancelled.

SKIPPED = 'SKIPPED'

Event is Skipped.

class AgilitySync.constants.DirectionTypes

This class defines constants that hold the direction types that a mapping can be configured in.

FORWARD = 'forward-direction'

To identify direction type during sync from which direction the event came

BACKWARD = 'backward-direction'

To identify direction type during sync from which direction the event came

class AgilitySync.constants.FieldTypes

This class defines constants that hold the different types of fields that Agility Sync recognizes, from the ALM systems.

TEXT = 'text'

This constant would be used if the field is text

URL = 'url'

This constant would be used if the field type is URL

DATE = 'date'

This constant would be used if the field type is date

DATETIME = 'datetime'

This constant would be used if the field type is datetime

TIME = 'time'

This constant would be used if the field type is time

BOOLEAN = 'bool'

This constant would be used if the field type is Boolean

BOOLEAN_LIST = 'boolean'

This constant would be used if the field type is Boolean list

USER = 'user'

This constant would be used if the field type is user

NUMERIC = 'numeric'

This constant would be used if the field type is numeric

LIST = 'list'

This constant would be used if the field type is list

TEAM = 'team'

This constant would be used if the field type is team

HTML = 'html'

This constant would be used if the field type is html

TEXTAREA = 'textarea'

This constant would be used if the field type is textarea

NONE = 'nofield'

This constant would be used if the field type is nofield

PROJECT = 'project'

This constant would be used if the field type is project

class AgilitySync.constants.FieldDisplayIcon

This class defines constants that identify the different icons for respective field types that are supported by AgilitySync.

DROPDOWN = 'DROPDOWN'

To the field as dropdown

CHECKBOX = 'CHECKBOX'

To the field as checkbox

NUMERIC = 'NUMERIC'

To the field as numeric

TEXT = 'TEXT'

To the field as text

RADIO_BUTTION = 'RADIO_BUTTION'

To the field as radio_button

USER = 'USER'

To the field as user

BOOLEAN = 'BOOLEAN'

To the field as Boolean

URL = 'URL'

To the field as URL

DATETIME = 'DATETIME'

To the field as datetime

DATE = 'DATE'

To the field as date

TEAM = 'TEAM'

To the field as team

HTML = 'HTML'

To the field as html

TIME = 'TIME'

To the field as time

TEXTAREA = 'TEXTAREA'

To the field as textarea

NO_FIELD = 'NO_FIELD'

To the field as no field

class AgilitySync.constants.AdditionalMapping

This class defines constants that are used to show the contents of additional mapping section of datamaps. These constants can be used in this API *AgilitySync.mapping.BaseField.fetch_supported_additional_mapping*

RELATIONSHIP = 'RELATIONSHIP'

Add this constant in if you want to display the field under Relationship section in UI

LINKS = 'LINKS'

Add this constant in if you want to display the field under Links section in UI

DEPENDENCY = 'DEPENDENCY'

Add this constant in if you want to display the field under Dependency section in UI

1.5 Exceptions —Agility SyncExceptions

class AgilitySync.exceptions.PayloadError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exception that arises at the level of parsing and analyzing the payload that is received from an ALM system's instance.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.EventError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exception that comes up when processing when creating the event from the payload received from the ALM system's instance.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SkipEvent(*reason*, *stack_trace=False*, *dont_log=False*)

This exception which is used to skip the event.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.InboundError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exception that arises at the level of processing the inbound event.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SkipFieldToNormalize(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to throw an exception when a field from the inbound should not be normalized.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.AttachmentDownloadError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exception that arises at the level of downloading an inbound attachment.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SkipAttachmentDownload(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to throw an exception when an attachment download must be skipped.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.OutboundError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exception that arises at the level of processing the outbound and syncing.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SkipOutbound(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to throw an exception when an outbound must be skipped.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SkipUpdate(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to throw an exception when an update must be skipped.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SkipFieldToSync(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to raise an exception when an outbound field must not be skipped from syncing.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.OutboundFieldError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exception that arises at the level of processing the outbound field for syncing.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.AttachmentUploadError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exception that arises at the level of uploading an attachment to the outbound ALM instance.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.WebhookError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class handles any exceptions that arise as part of creating the webhooks in the respective ALM system instances and the corresponding handlers on Agility Sync for accepting webhook data from them.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SkipWebhook(*reason*, *stack_trace=False*, *dont_log=False*)

This exception is raised when webhook creation needs to be skipped for a plugin.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.CustomizationError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to raise an exception when any errors are encountered during processing done by the customization framework if it is used for a particular mapping.

class AgilitySync.exceptions.TransformValueError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to raise an exception when something goes wrong while trying to transform a value.

class AgilitySync.exceptions.BlankFieldValueError(*reason*, *stack_trace=False*, *dont_log=False*) class AgilitySync.exceptions.SkipFieldException(*reason*, *stack_trace=False*, *dont_log=False*)

class AgilitySync.exceptions.SkipFieldFromMapping(*reason*, *stack_trace=False*, *dont_log=False*)

class AgilitySync.exceptions.TestConnectionError(*reason*, *stack_trace=False*, *dont_log=False*)

This exception class is used to throw an exception when something goes wrong with testing connection to a plugin instance.

Parameters

- reason (*str*) – reason for the exceptions.
- stack_trace (*bool*) – Log the trace if something went wrong.
- dont_log (*bool*) – Would not log the reason.

class AgilitySync.exceptions.SanitizedPluginError(*message*, *raw_details=None*)

A custom Exception type for Continuum Plugins which carries both a user-appropriate message (which doesn't reveal details about a foreign server, for example) and the nitty gritty details of the problem, logging the former at instantiation time at the "warning" level.

1.6 External Plugin

1.6.1 Introduction

Agility Sync supports third-party developed plugins too in addition to the plugins that are available as part of the Agility Sync itself. These third-party plugins must comply to a specific directory structure as explained, follow naming conventions expected and inherit specific classes and their methods in order to function as expected. The Agility Sync UI provides options to install a third-party plugin with its contents in zip format. It validates that the plugin complies to the expectations of the AS and if it does then the plugin will be installed and available for use along with the AS default plugins.

The following files are expected as part of the third-party plugin zip file.

- **config.json**
- **fieldtype_expression.json**
- **<LOGO_FILENAME>.png**

- **mapping.py**
- **sync.py**

1.6.2 Plugin Configuration

Agility Sync expects a config.json as part of the contents of the third-party plugin. This config.json will be used to determine what fields of what types should be displayed and taken inputs for when the user wants to create an instance of such an installed third-party plugin. The expectants are as listed below.

config.json

```
{
    "display_name": "The display name of the plugin.  This is different from the
→internal  name
                    and is usually more elaborate and indicative of the type of the
→plugin.",

    "description": "A brief description about the plugin.",

    "logo_file": "The file name of the image to be used as the display icon for this
→plugin.",

    "integration_type": "What type of plugin this is whether ALM or SCM or whatever.",

    "is_webhook_at_project_level": " - A Boolean to denote whether this plugin supports
                                    webhooks only at project level, where true means the
                                    support is only at project level and not system wide.
→",

    "instance_properties": [
        {
            "title": "Display name of field 1 on the UI for creating an instance of this
                      plugin type.",

            "name": "Internal name of field 1.",

            "type": "Type of field 1, which has to be input if you require the user to
→input a
                     value for this field.",

            "description": "A brief description about the field 1 which will be
→displayed if
                            the user hovers over it."
        },
        {
            "title": "Display name of field 2 on the UI for creating an instance of this
                      plugin type.",

            "name": "Internal name of field 2.",

            "type": "Type of field 1, which has to be input if you require the user to
→input  a
```

(Continues on next page)

(Continued from previous page)

```
                    value for this field.",

            "description": "A brief description about the field 2 which will be
→displayed if
                                the user hovers over it.",

            "format": "Must be set to password if field 2 is a password/token type field
→that
                                needs to be masked."
        }
    ],

    "assets_properties": [
        {
            "type": "Type of the first level of granularity in the ALM system, typically
                    this is project unless there are levels above them.",

            "title": "The display name of the field to be displayed when a mapping for an
                    instance of this plugin type is being created.",

            "name": "The internal name of the field.",

            "description": "A brief description about the category or granularity level.
→",

            "is_render_treeview": "A Boolean indicating whether tree view is required for
                                displaying the contents of this type.",

            "is_allow_duplicate_projects": "A Boolean to indicate whether duplicate
→selections
                                of values for this type is allowed.",

            "max_projects": "An integer representing the maximum number of selections of
                                values for this type that are allowed per mapping."
        },
        {
            "type": "Type of the second level of granularity in the ALM system,
→typically this
                    is asset/workitem unless there are levels above them.",

            "title": "The display name of the field to be displayed when a mapping for an
                    instance of this plugin type is being created.",

            "name": "The internal name of the field.",

            "description": "A brief description about the category or granularity level.
→",

            "dependent_assets": "An array listing all levels of granularities above this
→level
                                that this level is dependent on.  For example
→[projects]."
```

(Continues on next page)

(Continued from previous page)

```
        }
    ],

    "additional_mapping": {

        "attachment": {
            "enabled": "True/False - (A Boolean representing whether syncing of␣
↪attachments
                       are supported for this plugin type.)"
        },

        "comment": {
            "enabled": "A Boolean representing whether syncing of comments are supported␣
↪for
                       this plugin type."
        },

        "dependencies": {
            "enabled": "A Boolean representing whether syncing of dependencies are␣
↪supported
                       for this plugin type."
        },

        "relationship": {
            "enabled": "A Boolean representing whether syncing of relationships are␣
↪supported
                       for this plugin type."
        },

        "sync_reference": {
            "enabled": "A Boolean representing whether sync references creation in the␣
↪source
                       instance is supported for this plugin type."
        },

        "missing_events": {
            "enabled": "A Boolean representing whether syncing of events whose webhooks
                       weren't received are supported for this plugin type."
        },

        "data_migration": {
            "enabled": "A Boolean representing whether migration of data from an␣
↪instance is
                       supported for this plugin type."
        }
    }
}
```

1.6.3 FieldType Expression

The FieldType Expression must be provided by the plugin via a file named fieldtype_expression.json. This JSON file shall contain the payload pattern that when applied to the received payload from the ALM system instance via webhook, using jsonpath, shall return the value for a particular field. The definitions in this file are for each specific field type. For each field type an ALL_FIELDS key denotes the default pattern to apply for any field of that particular type unless the specific field's name has a definition on its own within the field type's definition. A sample fieldtype_expression.json file is given below for reference. A default definition may also be given which will be used in case the field's type does not have a definition in the file.

fieldtype_expression.json

```
{
    "list": {
        "$ALL_FIELDS$": {
            "multivalue": {
                "payload_pattern": {
                    "create":  "'$.changes[?(@.name==\\'%s\\')]' % (self.name)",
                    "update":  "'$.changes[?(@.name==\\'%s\\')]' % (self.name)"
                }
            }
        }
    },
    "user": {
        "$ALL_FIELDS$": {
            "multivalue": {
                "payload_pattern": {
                    "create":  "'$.changes[?(@.name==\\'%s\\')]' % (self.name)",
                    "update":  "'$.changes[?(@.name==\\'%s\\')]' % (self.name)"
                },
                "is_reset_value": false
            }
        },
        "CreatedBy": {
            "payload_pattern": {
                "create":  "$.snapshot"
            }
        }
    },
    "text": {
        "$ALL_FIELDS$": {
            "multivalue": {
                "payload_pattern": {
                    "create":  "'$.changes[?(@.name==\\'%s\\')]' % (self.name)",
                    "update":  "'$.changes[?(@.name==\\'%s\\')]' % (self.name)"
                }
            }
        },
        "Workitem_ID": {
            "payload_pattern": {
                "create":  "$.targetAsset._oid",
                "update":  "$.targetAsset._oid"
```

(Continues on next page)

(Continued from previous page)

```
            }
        },
        "workitem_display_id": {
            "payload_pattern": {
                "create":   "$.snapshot.Number",
                "update":   "$.snapshot.Number"
            }
        }
    },
    "url": {
        "Workitem_URL":  {
            "payload_pattern": {
                "create":   "$.targetAsset.href",
                "update":  "$.targetAsset.href"
            }
        }
    },
    "datetime": {
        "CreateDate": {
            "payload_pattern": {
                "create":   "$.snapshot.CreateDate"
            }
        },
        "ChangeDate": {
            "payload_pattern": {
                "update":   "$.snapshot.ChangeDate"
            }
        }
    },
    "default": {
        "payload_pattern": {
            "create":  "'$.changes[?(@.name==\\'%s\\').new]' % (self.name)",
            "update":   "'$.changes[?(@.name==\\'%s\\').new]'  %  (self.name)"
        },
        "blank_value":  "NULL",
        "reset_value": null
    }
}
```

1.6.4 Sync - Sample Sync Code

Agility Sync expects a sync.py file as part of the third-party plugin zip file. The expected classes and methods inside each class are to be inferred from the sample/example given below.

sync.py

```
from AgilitySync.sync import (
    as_exceptions,
    as_log,
    BaseInbound,
    BaseOutbound,
    BasePayload,
    BaseEvent,
    BaseAttachmentUpload,
    plugin_info,
    FieldTypes,
    EventCategory,
    EventTypes
)

class Payload(BasePayload):
    """
    This class must be implemented by the plugin by inheriting from the BasePayload class.
    It is expected to contain the methods to process the payload received from the ALM system.
    """

    def fetch_events(self) -> List[Dict]:
        """ `BasePayload.fetch_events`
        """

        return self.payload.get('events', [])

    def fetch_project(self, event) -> Union[int, str]:
        """
        Below is the sample code. Plugin must override this function
        """

        if event['webhookEvent'] in ('issuelink_created', 'issuelink_deleted'):
            RETURN event["project_id"]
        else:
            return event['issue']['fields']['project']['id']

    def fetch_asset(self, event) -> Union[str, int]:
        if event['webhookEvent'] in ('issuelink_created', 'issuelink_deleted'):
            return event["issuetype_id"]
        else:
            return event['issue']['fields']['issuetype']['id']

    def is_cyclic_event(self, event, sync_user) -> bool:
        if event['instigator']['_oid'] == sync_user_oid:
            as_log.warn("User [%s] is matched with Echo sync user [%s]" % (event['instigator']['_oid'], sync_user_oid))
            return True

        return False
```

(Continues on next page)

(Continued from previous page)

```
class Event(BaseEvent):

    def fetch_event_type(self) -> Union[str, int]:
        event_type = self.event['webhookEvent']
        if event_type in ('jira:issue_created', 'sprint_created'):
            return EventTypes.CREATE
        else:
            return EventTypes.UPDATE

    def fetch_workitem_id(self) -> Union[str, int]:
        return self.event['snapshot']['_oid']

    def fetch_workitem_display_id(self):
        return self.event['issue']['key']

    def fetch_workitem_url(self) -> str:
        return '/browse/{}'.format(self.event['issue']['key'])

    def fetch_revision(self):
        return self.event['snapshot']['_oid']

    def fetch_timestamp(self) -> datetime:
        timestamp = parser.parse(self.event['timestamp'])
        return datetime.fromtimestamp(time.mktime(timestamp.utctimetuple()))


class Inbound(BaseInbound):

    def connect(self):
        try:
            return _connect(self.instance_details['url'], self.instance_details['token'])
        except Exception as e:
            error_msg = 'Unable to connect to <plugin> host [{}]. The error is [{}].'
→format(self.instance_details['url'],

→str(e))
            raise as_exceptions.InboundError(error_msg, stack_trace=True)

    def fetch_event_category(self):
        category = []

        asset_type = self.event['targetAsset']['assetType']
        if asset_type == 'Attachment':
            category.append(EventCategory.ATTACHMENT)
        elif asset_type == 'Expression':
            category.append(EventCategory.COMMENT)
        else:
            category.append(EventCategory.WORKITEM)

        return category
```

(Continues on next page)

(Continued from previous page)

```
    def fetch_parent_id(self):
        if "relations" not in self.event.get('resource', {}):
            return None, None

        if self.event_type == EventTypes.CREATE:
            parent_id = self._get_relations(self.event['resource']['relations'])
            return parent_id, None
        elif self.event_type == EventTypes.UPDATE:
            parent_id, old_parent_id = self._get_relations_with_old(self.event['resource
→']['relations'])
            return parent_id, old_parent_id

    def fetch_comment(self):
        if self.event_type == EventTypes.CREATE:
            comment = self.event.get('resource', {}).get('fields', {}).get('System_
→History', None)
        else:
            comment = self.event.get('resource', {}).get('fields', {}).get('System_
→History', {}).get('newValue', None)

        return comment

    def fetch_attachments_metadata(self):
        attachment_doc = {
            "id": transformer_functions.unmoment_an_oid(self.event['snapshot']['_oid']),
            "headers": {
                "Authorization": self.instance_object.token
            }
        }
        for changes in self.event['changes']:
            name = changes['name']
            if name == 'Content':
                attachment_doc["url"] = "{}{}".format(self.instance_object.url,
                                                      re.sub('^\\/[^\\/]+', '', changes[
→'new']))
            elif name == 'ContentType':
                attachment_doc['content_type'] = changes['new']
            elif name == 'Filename':
                attachment_doc['filename'] = changes['new']
            elif name == 'AssetState':
                if changes['new'] == 'Active':
                    attachment_doc['type'] = 'ADDED'
                elif changes['new'] == 'Deleted':
                    attachment_doc['type'] = 'REMOVED'

        return [attachment_doc]

    def normalize_texttype_fieldvalue(self, field_value, field_attr):
        return self._normalize_field(field_value)

    def normalize_texttype_multivalue_field(self, field_value, field_attr):
```

(Continues on next page)

(Continued from previous page)

```
        multi_select_field_values = []

        for value in field_value:
            if value['act'] == 'Added':
                multi_select_field_values.append({'field_value': value['new'], 'act':
→'add'})
            else:
                multi_select_field_values.append({'field_value': value['new'], 'act':
→'remove'})

        return multi_select_field_values

    def normalize_usertype_fieldvalue(self, field_value, field_attr):
        return {"username": "<>", "email": "<>"}

    def normalize_usertype_multivalue_field(self, field_value, field_attr):
        return [
            "act": "add", "field_value": {"username": "testuser", "email":
→"testuser@test.com"
            "act": "remove", "field_value": {"username": "testuser1", "email":
→"testuser1@test.com"
        ]

    def create_remote_link(self, display_id, url, id_field, url_field):
        try:
            fields = {
                'Attributes': {}
            }

            if id_field:
                fields['Attributes'][id_field["name"]] = {'act': 'set', 'value': display_
→id}
            if url_field:
                fields['Attributes'][url_field["name"]] = {'act': 'set', 'value': url}

            if fields['Attributes']:
                self.instance_object._api_post(self.workitem_id.replace(':', '/'),␣
→fields, 'json', 'json')
            else:
                self.instance_object._push_link(self.workitem_id,  url,
                                                str(display_id))
        except Exception as e:
            as_log.warn('Unable to create remote link in Digital.ai Agility.  Error is [
→{}].'.format(e))


class Outbound(BaseOutbound):

    def connect(self):
        try:
            return _connect(self.instance_details['url'], self.instance_details['token'])
        except Exception as e:
```

(Continues on next page)

(Continued from previous page)

```
            error_msg = 'Unable to connect to V1 host [%s].  The error is [%s].' % (self.
→instance_details['url'], e)
            raise as_exceptions.OutboundError(error_msg, stack_trace=True)

    def create(self, sync_fields):
        asset_type = self.asset_info["asset"]
        try:
            response = self.instance_object._api_post(asset_type, sync_fields['fields'],
→'json', 'json')
            workitem_id = response['id']
            xref_object = {
                'id': workitem_id,
                'sync_info': sync_fields['fields']
            }
            xref_object['display_id'] = self.get_workitem_number(asset_type, workitem_id)
            xref_object['relative_url'] = '/{}.mvc/Summary?oidToken={}'.format(asset_
→type.lower(), workitem_id)
            xref_object["absolute_url"] = "{}{}".format(self.instance_details["url"].
→rstrip("/"),
                                                        xref_object["relative_url"])

            return xref_object
        except Exception as e:
            error_msg = ('Create [{}] failed in Digital.ai Agility. Trying to sync these␣
→fields \n[{}]\n.'
                         'Error is [{}].'.format(asset_type, sync_fields['fields'], e))
            raise as_exceptions.OutboundError(error_msg, stack_trace=True)

    def update(self, sync_fields):
        # local function

        self._update_issue(sync_fields)

    def comment_create(self, comment):
        try:
            self._issue_add_comment(self.instance_object, self.workitem_id, comment)
        except Exception as e:
            error_msg = "Unable to sync comment. Error is [{}]. The comment is [{}]".
→format(e, comment)
            raise as_exceptions.OutboundError(error_msg, stack_trace=True)

    def transform_fields(self, transfome_field_objs):
        fields = {}

        for outbound_field in transfome_field_objs:
            field_name = outbound_field.name
            field_value = outbound_field.value
            if outbound_field.is_multivalue:
                fields[field_name] = ",".join(field_value)
            else:
                fields[field_name] = field_value
```

(Continues on next page)

(Continued from previous page)

```
        sync_fields = {'fields': fields}

        return sync_fields

    def create_remote_link(self, workitem_id, workitem_url, id_field, url_field):
        try:
            fields = {
                'Attributes': {}
            }
            if id_field:
                fields['Attributes'][id_field["name"]] = {'act': 'set', 'value':␣
↪workitem_id}
            if url_field:
                fields['Attributes'][url_field["name"]] = {'act': 'set', 'value':␣
↪workitem_url}

            if fields['Attributes']:
                self.instance_object._api_post(self.workitem_id.replace(':', '/'),␣
↪fields, 'json', 'json')
            else:
                self.instance_object._push_link(self.workitem_id,  workitem_url,␣
↪str(workitem_id))
        except Exception as e:
            as_log.warn('Unable to create remote link in Digital.ai Agility.  Error is [
↪{}].'.format(e))

    def transform_texttype_multivalue(self, value, field_obj):
        # field level transform for multi text field

        if self.inbound.event_type == EventTypes.CREATE:
            return [{'act': val['act'], 'field_value': val['field_value'].replace(' ', '_
↪')} for val in value]
        elif value[0]['act'] == 'set':
            return [{'act': val['act'], 'field_value': val['field_value'][0].replace(' ',
↪ '_')} for val in value]
        else:
            return self._transform_multi_select_text_for_update(value,  field_obj.name)

    def transform_user(self, value, field_obj):
        # Internal function to transform the received user.

        return self._transform_single_select_user(value)


class AttachmentDownload(BaseAttachmentDownload):

    def download(self, storage_obj, verify_tls):
        # Plugin may override this function if plugin wants to download by own

        request_obj = requests.get(url=self.url, stream=True, headers=self.headers,
                                   verify=verify_tls, cookies=self.cookies)
        try:
```

(Continues on next page)

(Continued from previous page)

```
            with open(self.absolute_filename, 'wb') as base64_storage_obj:
                for data in request_obj.iter_content(self.chunk_size):
                    base64_storage_obj.write(data)

        except Exception as ex:
            raise  as_exceptions.AttachmentDownloadError(error_msg)


class AttachmentUpload(BaseAttachmentUpload):

    def upload(self, storage_obj, verify_tls):
        # Plugin may override this function if plugin wants to upload

        atch_obj = self.outbound.instance_object.add_attachment(self.workitem_id,␣
↪storage_obj)
        self.private_data['atch_obj'] = jira_atch_obj

    def fetch_id(self):
        return self.private_data['atch_obj'].id

    def fetch_url(self):
        return self.private_data['atch_obj'].content

    def remove(self, attachment_id, verify_tls):
        auth = HTTPBasicAuth(self.outbound.instance_details['user'], self.outbound.
↪instance_details['password'])
        remove_attachment_api = "{}/rest/api/2/attachment/{}".format(self.outbound.
↪instance_details['url'],
                                                                     attachment_id)
        r = requests.delete(remove_attachment_api, auth=auth, verify=verify_tls)
        try:
            r.raise_for_status()
        except requests.exceptions.HTTPError as ex:
            as_log.error("Unable to delete attachment. The error is %s" % r.content)
            raise  ex

    def fetch_header_info(self):
        return {"Authorization": self.instance_object.token}
```

1.6.5 Mapping - Sample mapping Code

Agility Sync expects a mapping.py file as part of the third-party plugin zip file. The expected classes and methods inside each class must be inferred from the sample/example given below.

mapping.py

```
from AgilitySync.mapping import (
    BaseField,
    BaseAssetsManage,
    BaseWebHook,
    as_exceptions,
    BaseFields,
    FieldTypes,
    FieldDisplayIcon,
    as_exceptions,
    as_log
)

class Field(BaseField):

    def is_required_field(self):
        if self.field_attr[1].get('required') is True:
            return True

        return False

    def is_disabled_field(self):
        return False

    def is_custom_field(self):
        if self.field_attr[1].get('schema').get('custom'):
            return True

        return False

    def is_readonly_field(self):
        if self.field_attr[1].get('readonly') is True:
            return True

        return False

    def fetch_name(self):
        return self.field_attr[0]

    def fetch_display_name(self):
        return self.field_attr[1].get('name', '')

    def fetch_fieldtype_info(self):
        if attribute_type == 'Text':
            return self._field_type_info(FieldTypes.URL, FieldDisplayIcon.URL)
        elif attribute_type == 'Numeric':
            return self._field_type_info(FieldTypes.NUMERIC, FieldDisplayIcon.NUMERIC)

    def is_multivalue_field(self):
        return self.field_attr[1].get('schema').get('type') == 'array'

    def is_support_default_value(self):
```

(Continues on next page)

(Continued from previous page)

```
        return self.field_attr[1].get('hasDefaultValue')

    def _field_type_info(self, field_type, display_image, values=None, value_type=None):
        field_type_doc = {"type": field_type}

        if values is not None and value_type is not None:
            field_type_doc["value_type"] = value_type
            field_type_doc["values"] = values

        if display_image:
            field_type_doc["display_icon"] = display_image

        return field_type_doc

    def fetch_supported_additional_mapping(self):
        additional_mapping = []

        if self.name in self.supported_fields:
            additional_mapping.append(AdditionalMapping.RELATIONSHIP)


class Fields(BaseFields):

    def fetch_fields(self):
        fields_list = []

        for field in self.instance_obj.get_fields(self.project_info['id'], self.asset_
→info["display_name"]):
            if field["referenceName"] not in self.IGNORED_FIELDS:
                field_with_other_attrs = self.instance_obj.get_field_other_attrs(self.
→project_info['id'],
                                                                                 field[
→"referenceName"])
                fields_list.append(dict(list(field.items()) + list(field_with_other_
→attrs.items())))

        return fields_list


class AssetsManage(BaseAssetsManage):

    def connect(self):
        try:
            return _connect(self.instance_details['url'], self.instance_details['token'])
        except Exception as e:
            error_msg = 'Unable to connect to V1 host [{}]. The error is [{}].'.
→format(self.instance_details['url'],
                                                                                 ␣
→str(e))
            raise as_exceptions.InboundError(error_msg, stack_trace=True)

    def fetch_sync_user(self):
```

(Continues on next page)

(Continued from previous page)

```
        try:
            response = self.instance_obj._api_get('Member?sel=Username&where=IsSelf=\
→'True\'', None)
            return response['Assets'][0]['id']
        except Exception as e:
            raise as_exceptions.SanitizedPluginError('Unable to get the username.',␣
→str(e))

    def fetch_projects(self):
        response = self.instance_obj._api_get('Scope', 'sel=Name,ClosedDate,Parent')
        projects = []

        for project in response.get('Assets', {}):
            # to fetch only active planning levels or projects
            closed_date = project['Attributes']['ClosedDate']['value']

            if closed_date is None:
                projects.append(
                    {
                        'id': project['id'],
                        "project": project["id"],
                        'display_name': project['Attributes']['Name']['value'],
                        'parent_display_name': project['Attributes']['Parent.Name'][
→'value']
                    }
                )

        projects.append(
            {
                "id": AssetsManage.NO_PROJECT['id'],
                "project": AssetsManage.NO_PROJECT["id"],
                "display_name": AssetsManage.NO_PROJECT["name"],
                "parent_display_name": None
            }
        )

        return projects

    def fetch_assets(self):
        asset_types = []

        response = self.instance_obj._api_get('AssetType', 'where=Base.Name="Workitem",
→"PrimaryWorkitem","BaseAsset"')
        project_ids = [proj['id'] for proj in self.query_params["projects"]]

        for asset in response.get('Assets', {}):
            name = asset['Attributes']['Name']['value']
            asset_types.append({'id': asset['id'], "asset": name, 'display_name': name})

        return asset_types

    def is_instance_supported(self):
```

(Continues on next page)

(Continued from previous page)

```
        response = self.instance_obj._meta_request('Story')

        return (True, None) if int(response['Version'].split(".")[0]) >= 20 else (False,
→"20.0")

    def test_connection(self):
        try:
            return self.instance_obj.test_connection()
        except Exception as ex:
            raise as_exceptions.SanitizedPluginError("Unknown error connecting to
→Digital.ai Agility.", ex.__str__())


class WebHook(BaseWebHook):

    def create_webhook(self, webhook_name, webhook_url, webhook_description):
        self.instance_obj.webhook_post(json.dumps(self._get_v1_webhook_data(webhook_name,
→ webhook_url,
                                                                            webhook_
→description)))
```

PYTHON MODULE INDEX

a

AgilitySync.constants, 34
AgilitySync.exceptions, 37
AgilitySync.mapping, 6