EPC Encoding Utils

Release 1.4

AAC Engineering

Jul 05, 2022

CONTENTS:

Library for encoding/decoding and representing GS1 Electronic Product Codes (EPCs).

CHAPTER

ONE

CONTRIBUTE

- Issue Tracker: https://github.com/AACEngineering/epcpy-tools/issues
- Source Code: https://github.com/AACEngineering/epcpy-tools

CHAPTER

TWO

LICENSE

The project is licensed under the Apache v2 License.

2.1 Getting Started

Installation requirements:

- Python 3.5 or above

To install:

```
pip install epc-encoding-utils
```

Examples:

See *Examples*.

2.2 EPC Schemes

2.2.1 GID

class epc.schemes.**GID**(**args*, ***kwargs*)

The General Identifier EPC scheme is independent of any specifications or identity scheme outside the EPC global Tag Data Standard.

General syntax: urn:epc:id:gid:ManagerNumber.ObjectClass.SerialNumber

Example: urn:epc:id:gid:95100000.12345.400

Parameters
epc (*str, optional*) – Hexadecimal EPC tag data

manager_number(*manager_number*)

Set the manager number data for the tag.

The General Manager Number identifies an organizational entity (essentially a company, manager or other organization) that is responsible for maintaining the numbers in subsequent fields – Object Class and Serial Number.

GS1 assigns the General Manager Number to an entity, and ensures that each General Manager Number is unique. Note that a General Manager Number is not a GS1 Company Prefix. A General Manager Number may only be used in GID EPCs.

Parameters
manager_number (*str, int*) – GS1 manager number.

Raises
- **ValueError** – Unable to convert string to an integer.
- **AttributeError** – Input outside valid range (0 to 268435455).

Returns
The GID tag object.

Return type
epc.schemes.GID

object_class(*object_class*)

Set the object class data for the tag.

The Object Class is used by an EPC managing entity to identify a class or "type" of thing. These object class numbers, of course, must be unique within each General Manager Number domain.

Parameters
object_class (*str, int*) – Numeric object class.

Raises
- **ValueError** – Unable to convert string to an integer.
- **AttributeError** – Input outside valid range (0 to 16777215).

Returns
The GID tag object.

Return type
epc.schemes.GID

serial_number(*serial_number*)

Set the serial number data for the tag.

The Serial Number code, or serial number, is unique within each object class. In other words, the managing entity is responsible for assigning unique, non-repeating serial numbers for every instance within each object class.

Parameters
serial_number (*str, int*) – Numeric serial number.

Raises
- **ValueError** – Unable to convert string to an integer.
- **AttributeError** – Input outside valid range (0 to 68719476735).

Returns
The GID tag object.

Return type
epc.schemes.GID

property tag_uri

Returns
The tag's URI.

Return type
str

property pure_identity_uri

Returns
The tag's pure identity URI.

Return type
str

property values

Returns

Dictionary containing:

- `size` (int): the tag's size in bits
- `manager_number` (int)
- `object_class` (int)
- `serial_number` (int)

decode_epc(*hex_string*)

Decode an encoded tag and populate this object's values from it.

Parameters
hex_string (*str*) – Tag data encoded as a hexadecimal string.

Raises

- **ValueError** – EPC scheme header does not match input.
- **ValueError** – Supplied `hex_string` bit length invalid.

check_fields()

Checks to make sure all components of the tag are present, including `size`, `manager_number`, `object_class` and `serial_number`.

Raises
AttributeError – If any components of the tag are missing.

Hint: To get the encoded tag data, use python's built in conversion methods:

```
>>> tag = GID()
>>> tag.manager_number(31231).object_class(11).serial_number(12)

# Hexadecimal
>>> hex(tag)
'0x3500079ff00000b00000000c'

# Binary
>>> bin(tag)

↪'0b11010100000000000001111001111111110000000000000000000010110000000000000000000000000000000001100
↪'
```

Constants

GID.**SIZE_96**

96 bit tag size. Only size supported for this scheme.

GID.**HEADER_96**

GS1 specified hexadecimal header for 96 bit GID tags: `0x35`.

GID.**GID_96**

Human readable GS1 specified header for 96 bit GID tags: `gid-96`.

2.2.2 GIAI

class epc.schemes.**GIAI**(**args*, ***kwargs*)

The Global Individual Asset Identifier EPC scheme is used to assign a unique identity to a specific asset, such as a forklift or a computer.

The scheme supports two sizes: 96 bit and 202 bit tags. Alphanumeric character are supported on the larger size tag for the asset reference.

General syntax: urn:epc:id:giai:CompanyPrefix.IndividualAssetReference

Example: urn:epc:id:giai:0614141.1234540

Parameters

- **epc** (*str, optional*) – Hexadecimal EPC tag data
- **barcode** (*str, optional*) – GIAI barcode data
- **company_prefix_length** (*int, optional*) – Number of digits in the company prefix. Required when specifying a barcode.

filter(*tag_filter*)

The filter value is additional control information that may be included in the EPC memory bank of a Gen 2 tag. The intended use of the filter value is to allow an RFID reader to select or deselect the tags corresponding to certain physical objects, to make it easier to read the desired tags in an environment where there may be other tags present in the environment. For example, if the goal is to read the single tag on a pallet, and it is expected that there may be hundreds or thousands of item-level tags present, the performance of the capturing application may be improved by using the Gen 2 air interface to select the pallet tag and deselect the item-level tags.

Allowed values for GIAI tags:

Value	Constant	Description
0	FILTER_ALL	All Others
1	FILTER_RAIL	Rail Vehicle
2-7	FILTER_RESERVED_*	Reserved

Parameters
tag_filter (*int*) – The filter value, defaults to `FILTER_ALL`.

Raises
AttributeError – Filter must be between 0 and 7.

Returns
The GIAI tag object.

Return type
epc.schemes.GIAI

company_prefix(*company_prefix*, *company_prefix_length=None*)

The GS1 Company Prefix, assigned by GS1 to a managing entity. The Company Prefix is the same as the GS1 Company Prefix digits within a GS1 GIAI key.

Length corresponds to the number of digits in the company prefix.

Parameters

- **company_prefix** (*str, int*) – GS1 company prefix.
- **company_prefix_length** (*int, optional*) – Number of digits in the company prefix. Required when company_prefix is an int.

Returns
The GIAI tag object.

Return type
epc.schemes.GIAI

asset_reference(*asset_reference*)

The Individual Asset Reference, assigned uniquely by the managing entity to a specific asset.

Parameters
asset_reference (*str, int*) – The asset reference.

Raises

- **AttributeError** – Reference length must be less than 24 characters.
- **ValueError** – Reference string character not encodeable.

Returns
The GIAI tag object.

Return type
epc.schemes.GIAI

tag_size()

Set the size for the tag. Options are SIZE_96 or SIZE_202.

Parameters
tag_size (*int*) – Tag size in bits. Defaults to SIZE_96.

Raises
AttributeError – Invalid tag size specified.

Returns
The GIAI tag object.

Return type
epc.schemes.GIAI

property tag_uri

Returns
The tag's URI.

Return type
str

property pure_identity_uri

Returns
The tag's pure identity URI.

Return type
str

property barcode

Returns
The barcode representation of the tag.

Return type
str

property barcode_humanized

Returns
A human readable barcode representation of the tag.

Return type
str

property values

Returns

Dictionary containing:

- `size` (int): the tag's size in bits
- `filter` (int)
- `company_prefix` (str)
- `asset_reference` (int *or* str)

decode_epc(*hex_string*)

Decode an encoded tag and populate this object's values from it.

Parameters
hex_string (*str*) – Tag data encoded as a hexadecimal string.

Raises

- **ValueError** – EPC scheme header does not match input.
- **ValueError** – Filter does not match allowed values.
- **ValueError** – Supplied `hex_string` bit length invalid.

decode_barcode(*barcode*, *company_prefix_length*)

Decode a barcode and populate this object's values from it.

Parameters

- **hex_string** (*str*) – Barcode
- **company_prefix_length** (*int*) – Number of digits of the company prefix length

Raises

- **ValueError** – Expected barcode header does not match input.
- **AttributeError** – Invalid barcode length, or wrong company prefix.

check_fields()

Checks to make sure all components of the tag are present, including `size`, `filter`, `company_prefix` and `asset_reference`.

Raises

AttributeError – If any components of the tag are missing.

Hint: To get the encoded tag data, use python's built in conversion methods:

```
>>> tag = GIAI()
>>> tag.company_prefix('000200').asset_reference(50)

# Hexadecimal
>>> hex(tag)
'0x34180032000000000000032'

# Binary
>>> bin(tag)

↪'0b11010000011000000000000011001000000000000000000000000000000000000000000000000000000000000000000000000000110010
↪'
```

Constants

Filters

GIAI.FILTER_ALL

GIAI.FILTER_RAIL

GIAI.FILTER_RESERVED_*

Replace asterisk with 2-7.

Sizes

GIAI.SIZE_96

96 bit tag size.

GIAI.SIZE_202

202 bit tag size. Supports alphanumeric characters for the asset reference.

Headers

GIAI.HEADER_BARCODE

GS1 specified barcode header: `8004`.

GIAI.HEADER_96

GS1 specified hexadecimal header for 96 bit GIAI tags: `0x34`.

GIAI.HEADER_202

GS1 specified hexadecimal header for 202 bit GIAI tags: `0x38`.

GIAI.GIAI_96

Human readable GS1 specified header for 96 bit GIAI tags: `giai-96`.

GIAI.**GIAI_202**

Human readable GS1 specified header for 202 bit GIAI tags: `giai-202`.

2.2.3 GRAI

class epc.schemes.**GRAI**(**args*, ***kwargs*)

The Global Returnable Asset Identifier EPC scheme is used to assign a unique identity to a specific returnable asset, such as a reusable shipping container or a pallet skid.

General syntax: urn:epc:id:grai:CompanyPrefix.AssetType.SerialNumber

Example: urn:epc:id:grai:0614141.12345.400

filter(*tag_filter*)

The filter value is additional control information that may be included in the EPC memory bank of a Gen 2 tag. The intended use of the filter value is to allow an RFID reader to select or deselect the tags corresponding to certain physical objects, to make it easier to read the desired tags in an environment where there may be other tags present in the environment. For example, if the goal is to read the single tag on a pallet, and it is expected that there may be hundreds or thousands of item-level tags present, the performance of the capturing application may be improved by using the Gen 2 air interface to select the pallet tag and deselect the item-level tags.

Allowed values for GRAI tags:

Value	Constant	Description
0	FILTER_ALL	All Others
1-7	FILTER_RESERVED_*	Reserved

Parameters
tag_filter (*int*) – The filter value, defaults to `FILTER_ALL`.

Raises
AttributeError – Filter must be between 0 and 7.

Returns
The GRAI tag object.

Return type
epc.schemes.GRAI

company_prefix(*company_prefix*, *company_prefix_length=None*)

The GS1 Company Prefix, assigned by GS1 to a managing entity.

Length corresponds to the number of digits in the company prefix.

Parameters

- **company_prefix** (*str, int*) – GS1 company prefix.
- **company_prefix_length** (*int, optional*) – Number of digits in the company prefix. Required when `company_prefix` is an int.

Returns
The GRAI tag object.

Return type
epc.schemes.GRAI

asset_type(*asset_type*)

The Asset Type, assigned by the managing entity to a particular class of asset.

Parameters
asset_type (*int, str*) – The asset type.

Raises
ValueError – Unable to convert input to an integer.

Returns
The GRAI tag object.

Return type
epc.schemes.GRAI

serial_number(*serial_number*)

The Serial Number, assigned by the managing entity to an individual object. Because an EPC always refers to a specific physical object rather than an asset class, the serial number is mandatory in the GRAI-EPC.

Parameters
serial_number (*str, int*) – The serial number.

Raises

- **AttributeError** – Serial number bit length incorrect.
- **AttributeError** – Serial number length incorrect.

Returns
The GRAI tag object.

Return type
epc.schemes.GRAI

tag_size()

Set the size for the tag. Options are SIZE_96 or SIZE_170.

Parameters
tag_size (*int*) – Tag size in bits. Defaults to SIZE_96.

Raises
AttributeError – Invalid tag size specified.

Returns
The GRAI tag object.

Return type
epc.schemes.GRAI

property tag_uri

Returns
The tag's URI.

Return type
str

property pure_identity_uri

Returns
The tag's pure identity URI.

Return type
str

property barcode

Returns
The barcode representation of the tag.

Return type
str

property barcode_humanized

Returns
A human readable barcode representation of the tag.

Return type
str

property values

Returns

Dictionary containing:

- `size` (int): the tag's size in bits
- `filter` (int)
- `company_prefix` (str)
- `asset_type` (int)
- `serial_number` (int *or* str)

decode_epc(*hex_string*)

Decode an encoded tag and populate this object's values from it.

Parameters
hex_string (*str*) – Tag data encoded as a hexadecimal string.

Raises

- **ValueError** – EPC scheme header does not match input.
- **ValueError** – Filter does not match allowed values.
- **ValueError** – Supplied `hex_string` bit length invalid.

decode_barcode(*barcode*, *company_prefix_length*)

Decode a barcode and populate this object's values from it.

Parameters

- **hex_string** (*str*) – Barcode
- **company_prefix_length** (*int*) – Number of digits of the company prefix length

Raises

- **ValueError** – Expected barcode header does not match input.
- **AttributeError** – Invalid barcode length, or wrong company prefix.

check_fields()

Checks to make sure all components of the tag are present, including `size`, `filter`, `company_prefix`, `asset_type` and `serial_number`.

Raises
AttributeError – If any components of the tag are missing.

Hint: To get the encoded tag data, use python's built in conversion methods:

```
>>> tag = GRAI().tag_size(GRAI.SIZE_170)
>>> tag.company_prefix('000123').asset_type(8).serial_number('ABC')

# Hexadecimal
>>> hex(tag)
'0x3718001ec0000220c2860000000000000000000000'

# Binary
>>> bin(tag)

↪'0b1101110001100000000000000011110110000000000000000000000010001000001100001010000110000000000000000000000000
↪'
```

Constants

Filters

GRAI.**FILTER_ALL**

GRAI.FILTER_RESERVED_*

Replace asterisk with 1-7.

Sizes

GRAI.**SIZE_96**

96 bit tag size.

GRAI.**SIZE_170**

170 bit tag size.

Headers

GRAI.**HEADER_BARCODE**

GS1 specified barcode header: `8003`.

GRAI.**HEADER_96**

GS1 specified hexadecimal header for 96 bit GRAI tags: `0x33`.

GRAI.**HEADER_170**

GS1 specified hexadecimal header for 202 bit GRAI tags: `0x37`.

GRAI.**GRAI_96**

Human readable GS1 specified header for 96 bit GRAI tags: `grai-96`.

GRAI.**GRAI_170**

Human readable GS1 specified header for 202 bit GRAI tags: `grai-170`.

2.2.4 SGLN

class epc.schemes.**SGLN**(**args*, ***kwargs*)

The SGLN EPC scheme is used to assign a unique identity to a physical location, such as a specific building or a specific unit of shelving within a warehouse.

General syntax: urn:epc:id:sgln:CompanyPrefix.LocationReference.Extension

Example: urn:epc:id:sgln:0614141.12345.400

filter(*tag_filter*)

The filter value is additional control information that may be included in the EPC memory bank of a Gen 2 tag. The intended use of the filter value is to allow an RFID reader to select or deselect the tags corresponding to certain physical objects, to make it easier to read the desired tags in an environment where there may be other tags present in the environment. For example, if the goal is to read the single tag on a pallet, and it is expected that there may be hundreds or thousands of item-level tags present, the performance of the capturing application may be improved by using the Gen 2 air interface to select the pallet tag and deselect the item-level tags.

Allowed values for SGLN tags:

Value	Constant	Description
0	FILTER_ALL	All Others
1-7	FILTER_RESERVED_*	Reserved

Parameters
tag_filter (*int*) – The filter value, defaults to `FILTER_ALL`.

Raises
AttributeError – Filter must be between 0 and 7.

Returns
The SGLN tag object.

Return type
epc.schemes.SGLN

company_prefix(*company_prefix*, *company_prefix_length=None*)

The GS1 Company Prefix, assigned by GS1 to a managing entity. The Company Prefix is the same as the GS1 Company Prefix digits within a GS1 GIAI key.

Length corresponds to the number of digits in the company prefix.

Parameters

- **company_prefix** (*str, int*) – GS1 company prefix.
- **company_prefix_length** (*int, optional*) – Number of digits in the company prefix. Required when `company_prefix` is an int.

Returns
The SGLN tag object.

Return type
epc.schemes.SGLN

location_reference(*location_reference*)

The Location Reference, assigned uniquely by the managing entity to a specific physical location.

Parameters
location_reference (*int, str*) – The location reference

Raises
ValueError – Unable to convert input to an integer.

Returns
The SGLN tag object.

Return type
epc.schemes.SGLN

extension(*extension*)

The GLN Extension, assigned by the managing entity to an individual unique location. If the entire GLN Extension is just a single zero digit, it indicates that the SGLN stands for a GLN, without an extension.

Parameters
extension (*str, int*) – The GLN extension

Raises
AttributeError – Extension length incorrect.

Returns
The SGLN tag object.

Return type
epc.schemes.SGLN

tag_size()

Set the size for the tag. Options are SIZE_96 or SIZE_195.

Parameters
tag_size (*int*) – Tag size in bits. Defaults to SIZE_96.

Raises
AttributeError – Invalid tag size specified.

Returns
The SGLN tag object.

Return type
epc.schemes.SGLN

property tag_uri

Returns
The tag's URI.

Return type
str

property pure_identity_uri

Returns
The tag's pure identity URI.

Return type
str

property barcode

Returns
The barcode representation of the tag.

Return type
str

`property barcode_humanized`

Returns
A human readable barcode representation of the tag.

Return type
str

`property values`

Returns

Dictionary containing:

- `size` (int): the tag's size in bits
- `filter` (int)
- `company_prefix` (str)
- `location_reference` (int)
- `extension` (int *or* str)

`decode_epc`(*hex_string*)

Decode an encoded tag and populate this object's values from it.

Parameters
hex_string (*str*) – Tag data encoded as a hexadecimal string.

Raises

- **ValueError** – EPC scheme header does not match input.
- **ValueError** – Filter does not match allowed values.
- **ValueError** – Supplied `hex_string` bit length invalid.

`decode_barcode`(*barcode*, *company_prefix_length*)

Decode a barcode and populate this object's values from it.

Parameters

- **hex_string** (*str*) – Barcode
- **company_prefix_length** (*int*) – Number of digits of the company prefix length

Raises

- **ValueError** – Expected barcode header does not match input.
- **AttributeError** – Invalid barcode length, or wrong company prefix.

`check_fields`()

Checks to make sure all components of the tag are present, including `size`, `filter`, `company_prefix`, `location_reference` and `extension`.

Raises
AttributeError – If any components of the tag are missing.

Hint: To get the encoded tag data, use python's built in conversion methods:

```
>>> tag = SGLN()
>>> tag.company_prefix(1234, company_prefix_length=6).location_reference(15).
↪extension(1000)

# Hexadecimal
>>> hex(tag)
'0x3218013480001e00000003e8'

# Binary
>>> bin(tag)

↪'0b11001000011000000000001001101001000000000000000000001111000000000000000000000000000000000001111101000
↪'
```

Constants

Filters

SGLN.**FILTER_ALL**

SGLN.FILTER_RESERVED_*
: Replace asterisk with 1-7.

Sizes

SGLN.**SIZE_96**
: 96 bit tag size.

SGLN.**SIZE_195**
: 195 bit tag size.

Headers

SGLN.**HEADER_BARCODE**
: GS1 specified barcode header: `414`.

SGLN.**HEADER_96**
: GS1 specified hexadecimal header for 96 bit SGLN tags: `0x32`.

SGLN.**HEADER_195**
: GS1 specified hexadecimal header for 202 bit SGLN tags: `0x39`.

SGLN.**SGLN_96**
: Human readable GS1 specified header for 96 bit SGLN tags: `sgln-96`.

SGLN.**SGLN_195**
: Human readable GS1 specified header for 202 bit SGLN tags: `sgln-195`.

2.2.5 SGTIN

class epc.schemes.**SGTIN**(**args*, ***kwargs*)

The Serialised Global Trade Item Number EPC scheme is used to assign a unique identity to an instance of a trade item, such as a specific instance of a product or SKU.

General syntax: urn:epc:id:sgtin:CompanyPrefix.ItemRefAndIndicator.SerialNumber

Example: urn:epc:id:sgtin:0614141.112345.400

filter(*tag_filter*)

The filter value is additional control information that may be included in the EPC memory bank of a Gen 2 tag. The intended use of the filter value is to allow an RFID reader to select or deselect the tags corresponding to certain physical objects, to make it easier to read the desired tags in an environment where there may be other tags present in the environment. For example, if the goal is to read the single tag on a pallet, and it is expected that there may be hundreds or thousands of item-level tags present, the performance of the capturing application may be improved by using the Gen 2 air interface to select the pallet tag and deselect the item-level tags.

Allowed values for SGTIN tags:

Value	Constant	Description
0	FILTER_ALL	All Others
1	FILTER_POS	Point of Sale Trade Item
2	FILTER_FULL_CASE	Full Case for Transport
3	FILTER_RESERVED_3	Reserved
4	FILTER_INNER_PACK	Inner Pack Trade Item
5	FILTER_RESERVED_5	Reserved
6	FILTER_UNIT_LOAD	Unit Load
7	FILTER_INNER_ITEM	Unit Inside Trade Item

Parameters
tag_filter (*int*) – The filter value, defaults to `FILTER_ALL`.

Raises
AttributeError – Filter must be between 0 and 7.

Returns
The SGTIN tag object.

Return type
epc.schemes.SGTIN

company_prefix(*company_prefix*, *company_prefix_length=None*)

The GS1 Company Prefix, assigned by GS1 to a managing entity.

Length corresponds to the number of digits in the company prefix.

Parameters

- **company_prefix** (*str, int*) – GS1 company prefix.
- **company_prefix_length** (*int, optional*) – Number of digits in the company prefix. Required when `company_prefix` is an int.

Returns
The SGTIN tag object.

Return type
epc.schemes.SGTIN

`item_reference`(*item_reference*)

The Item Reference is assigned by the managing entity to a particular object class.

The Item Reference as it appears in the EPC URI is derived from the GTIN by concatenating the Indicator Digit of the GTIN (or a zero pad character, if the EPC URI is derived from a GTIN-8, GTIN-12, or GTIN-13) and the Item Reference digits, and treating the result as a single numeric string.

Parameters
item_reference (*int, str*) – The item reference.

Raises
ValueError – item_reference must be an integer

Returns
The SGTIN tag object.

Return type
epc.schemes.SGTIN

`serial_number`(*serial_number*)

The Serial Number, assigned by the managing entity to an individual object. The serial number is not part of the GTIN, but is formally a part of the SGTIN.

Parameters
serial_number (*str, int*) – The serial number.

Raises

- **AttributeError** – Serial number bit length incorrect.
- **AttributeError** – Serial number length incorrect.

Returns
The SGTIN tag object.

Return type
epc.schemes.SGTIN

`tag_size`()

Set the size for the tag. Options are `SIZE_96` or `SIZE_198`.

Parameters
tag_size (*int*) – Tag size in bits. Defaults to `SIZE_96`.

Raises
AttributeError – Invalid tag size specified.

Returns
The SGTIN tag object.

Return type
epc.schemes.SGTIN

`property tag_uri`

Returns
The tag's URI.

Return type
str

property pure_identity_uri

Returns
The tag's pure identity URI.

Return type
str

property barcode

Returns
The barcode representation of the tag.

Return type
str

property barcode_humanized

Returns
A human readable barcode representation of the tag.

Return type
str

property gtin

A Global Trade Item Number (GTIN) is the 14 digit GS1 Identification Key used to identify products. The key comprises a GS1 Company Prefix followed by an Item Reference Number and a Check Digit.

Returns
The GTIN-14 representation of the tag.

Return type
str

property values

Returns

Dictionary containing:

- `size` (int): the tag's size in bits
- `filter` (int)
- `company_prefix` (str)
- `item_reference` (int)
- `serial_number` (int *or* str)

decode_epc(*hex_string*)

Decode an encoded tag and populate this object's values from it.

Parameters
hex_string (*str*) – Tag data encoded as a hexadecimal string.

Raises

- **ValueError** – EPC scheme header does not match input.
- **ValueError** – Filter does not match allowed values.
- **ValueError** – Supplied `hex_string` bit length invalid.

decode_barcode(*barcode*, *company_prefix_length*)

Decode a barcode and populate this object's values from it.

Parameters

- **hex_string** (*str*) – Barcode
- **company_prefix_length** (*int*) – Number of digits of the company prefix length

Raises

- **ValueError** – Expected barcode header does not match input
- **AttributeError** – Invalid check digit
- **AttributeError** – Invalid barcode length, or wrong company prefix
- **AttributeError** – Invalid item_reference in barcode

decode_gtin(*gtin*, *company_prefix_length*, *serial_number=0*)

Decode a GTIN (Supports GTIN-14, GTIN-13, GTIN-12 formats) to populate this object's values from it. Check digit must be included.

Parameters

- **gtin** (*str*) – Global trade item number
- **company_prefix_length** (*int*) – Number of digits of the company prefix length
- **serial_number** (*int, str, optional*) – Serial number to set on the tag. Defaults to *0*.

Raises

- **ValueError** – Invalid GTIN length
- **AttributeError** – Invalid check digit
- **AttributeError** – Invalid company_prefix_length specified
- **AttributeError** – Invalid item_reference in gtin

check_fields()

Checks to make sure all components of the tag are present, including `size`, `filter`, `company_prefix`, `item_reference` and `serial_number`.

Raises

AttributeError – If any components of the tag are missing.

Hint: To get the encoded tag data, use python's built in conversion methods:

```
>>> tag = SGTIN().tag_size(SGTIN.SIZE_198)
>>> tag.company_prefix('000123').item_reference(18).serial_number('ABC')

# Hexadecimal
>>> hex(tag)
'0x3618001ec00004a0c2860000000000000000000000000000000000'

# Binary
>>> bin(tag)

↪'0b11011000001100000000000000011111011000000000000000000000100101000001100001010000110000000000000000000000000
↪'
```

Constants

Filters

SGTIN.**FILTER_ALL**

SGTIN.**FILTER_POS**

SGTIN.**FILTER_FULL_CASE**

SGTIN.**FILTER_RESERVED_3**

SGTIN.**FILTER_INNER_PACK**

SGTIN.**FILTER_RESERVED_5**

SGTIN.**FILTER_UNIT_LOAD**

SGTIN.**FILTER_INNER_ITEM**

Sizes

SGTIN.**SIZE_96**
: 96 bit tag size.

SGTIN.**SIZE_198**
: 198 bit tag size.

Headers

SGTIN.**HEADER_BARCODE**
: GS1 specified GTIN barcode header: `01`.

SGTIN.**HEADER_BARCODE_SERIAL_NUMBER**
: GS1 specified GTIN serial barcode header: `21`.

SGTIN.**HEADER_96**
: GS1 specified hexadecimal header for 96 bit SGTIN tags: `0x30`.

SGTIN.**HEADER_198**
: GS1 specified hexadecimal header for 202 bit SGTIN tags: `0x36`.

SGTIN.**SGTIN_96**
: Human readable GS1 specified header for 96 bit SGTIN tags: `sgtin-96`.

SGTIN.**SGTIN_198**
: Human readable GS1 specified header for 198 bit SGTIN tags: `sgtin-198`.

2.3 Utilities

epc.utils.**decode_epc**(*hex_string*)

Attempt to decode a hex string to an EPC tag. Returns a tag object if successful.

Parameters
epc (*str*) – Hexadecimal EPC tag data

Raises

- **NotImplementedError** – Unable to determine tag encoding.
- **NotImplementedError** – Scheme not implemented for tag.

Returns
EPC tag object

Return type
object

epc.utils.**get_epc_encoding**(*hex_string*)

Determine the encoding used on the provided tag.

Parameters
epc (*str*) – Hexadecimal EPC tag data

Raises
LookupError – Unable to match encoding.

Returns
Matching EPC scheme

Return type
class

epc.utils.barcode.**decode_barcode**(*barcode_string*, *company_prefix_length*)

Attempt to decode a barcode to an EPC tag. Returns a tag object if successful.

Parameters

- **barcode_string** (*str*) – Barcode data
- **company_prefix_length** (*int*) – Number of digits in the company prefix

Raises

- **NotImplementedError** – Unable to determine tag encoding.
- **NotImplementedError** – Scheme not implemented for barcode.

Returns
EPC tag object

Return type
object

2.4 Examples

2.4.1 General

Decode an EPC with unknown encoding

```
>>> from epc.utils import decode_epc
>>> decode_epc('3414013880000000000000001')  # '0x' prefix is optional
<epc.schemes.GIAI urn:epc:id:giai:0020000.1>
```

Decode a barcode

```
>>> from epc.utils.barcode import decode_barcode
>>> decode_barcode('80030000000100000141', company_prefix_length=6)
<epc.schemes.GRAI urn:epc:id:grai:000001.000001.1>
```

Tip: The *company_prefix_length* field is the number of digits in the expected GS1 Company Prefix.

Convert an EPC to barcode (for schemes that support barcodes)

```
>>> from epc.utils import decode_epc
>>> decode_epc('3414013880000000000000001').barcode
'800400200001'
```

2.4.2 GID Tags

epc.schemes.GID

```
>>> from epc.schemes import GID

# Create an uninitialized tag object
>>> my_tag = GID()

# Assign the required tag attributes
>>> my_tag.manager_number(951001).object_class(15).serial_number(1)
<epc.schemes.GID urn:epc:id:gid:951001.15.1>

# Get the encoded tag data
>>> hex(my_tag)
'0x3500e82d900000f000000001'

>>> my_tag.pure_identity_uri
'urn:epc:tag:gid-96:951001.15.1'

# Strings will be automatically converted to integers
>>> my_tag.serial_number('90')
<epc.schemes.GID urn:epc:id:gid:951001.15.90>

# Initialize a new tag from tag data
>>> GID(epc='0x3500e82d900000f000000001')
<epc.schemes.GID urn:epc:id:gid:951001.15.1>
```

2.4.3 GIAI Tags

epc.schemes.GIAI

```
>>> from epc.schemes import GIAI

>>> my_tag = GIAI()

# Initialize tag
>>> my_tag.company_prefix('000200').asset_reference(50)
<epc.schemes.GIAI urn:epc:id:giai:000200.50>

# Set tag size to 202 bit
>>> my_tag.tag_size(GIAI.SIZE_202)
>>> my_tag.asset_reference('HIPPO')
<epc.schemes.GIAI urn:epc:id:giai:000200.HIPPO>

# Get the barcode
>>> my_tag.barcode
'8004000200HIPPO'

>>> my_tag.barcode_humanized
'(8004) 000200 HIPPO'

# Set a tag filter
>>> my_tag.filter(GIAI.FILTER_RAIL)
```

2.4.4 GRAI Tags

epc.schemes.GIAI

```
>>> from epc.schemes import GRAI

>>> my_tag = GRAI().tag_size(GRAI.SIZE_170)

>>> my_tag.company_prefix('000123').asset_type(8).serial_number('WOW!')
<epc.schemes.GRAI urn:epc:id:grai:000123.000008.WOW!>

# Get tag URI
>>> my_tag.tag_uri
'urn:epc:tag:grai-170:0.000123.000008.WOW!'

# Barcode
>>> my_tag.barcode
'80030000012300000082WOW!'
```

2.4.5 SGLN Tags

epc.schemes.SGLN

```
>>> from epc.schemes import SGLN

>>> my_tag = SGLN()

>>> my_tag.company_prefix('001234').location_reference(15).extension(1000)
<epc.schemes.SGLN urn:epc:id:sgln:001234.000015.1000>
```

2.4.6 SGTIN Tags

epc.schemes.SGTIN

```
>>> from epc.schemes import SGTIN

>>> my_tag = SGTIN()

>>> my_tag.company_prefix('001234').item_reference(15).serial_number(1000)
<epc.schemes.SGTIN urn:epc:id:sgtin:001234.0000015.1000>

# Create a tag from a GTIN
>>> my_tag = SGTIN()
>>> my_tag.decode_gtin('80614141123458', company_prefix_length=7, serial_number=6789)
>>> my_tag.tag_uri
'urn:epc:tag:sgtin-96:0.0614141.812345.6789'
```

PYTHON MODULE INDEX

e

INDEX

L

M

O

P

S

T

V