Since Tether's launch in 2014, stablecoins have grown to become one of the most dominant sectors in the world of digital assets. Many considered them to be crypto's "killer app," providing global access to stable savings and payment rails beyond the rigid confines of the traditional banking system. In this walkthrough, we use Coin Metrics **Market Data Feed** and **CM Prices** to explore the various venues where these assets are traded, and examine various pricing methodologies that allow us to better understand how they perform in comparison to the underlying fiat currencies.

Resources

This notebook demonstrates basic functionality offered by the Coin Metrics Python API Client and Market Data Feed.

Coin Metrics offers a vast assortment of data for hundreds of cryptoassets. The Python API Client allows for easy access to this data using Python without needing to create your own wrappers using `requests` and other such libraries.

To understand the data that Coin Metrics offers, feel free to peruse the resources below.

- The Coin Metrics API v4 website contains the full set of endpoints and data offered by Coin Metrics.
- The Coin Metrics Knowledge Base gives detailed, conceptual explanations of the data that Coin Metrics offers.
- The API Spec contains a full list of functions.

Notebook Setup

In [50]:
```
from os import environ
import sys
import pandas as pd
import seaborn as sns
import logging
from datetime import date, datetime, timedelta
from coinmetrics.api_client import CoinMetricsClient
import json
import logging
from pytz import timezone as timezone_conv
from datetime import timezone as timezone_info
import matplotlib
import matplotlib.dates as mdates
from matplotlib.dates import MonthLocator, DateFormatter, YearLocator, AutoDateLocator
from matplotlib.ticker import NullFormatter
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np
%matplotlib inline
```

In [65]:
```
sns.set_theme()
plt.rcParams.update({'font.size': 16, 'font.family': 'Lato'})
sns.set(rc={'figure.figsize':(14,8)})
```

In [3]:
```
logging.basicConfig(
    format='%(asctime)s %(levelname)-8s %(message)s',
    level=logging.INFO,
    datefmt='%Y-%m-%d %H:%M:%S'
)
```

In [4]:
```
# We recommend privately storing your API key in your local environment.
try:
    api_key = environ["CM_API_KEY"]
    logging.info("Using API key found in environment")
except KeyError:
    api_key = ""
    logging.info("API key not found. Using community client")

client = CoinMetricsClient(api_key)
```

```
2023-05-17 10:23:57 INFO     Using API key found in environment
```

Get Stablecoin Markets

The catalog/markets endpoint returns a list of available markets along with time ranges of available data. Users can pass in a list of markets, exchanges, or market types (spot, futures, options). We can retrieve our stablecoin markets by fetching a list of all 'spot' markets, then filtering for the markets where the 'base' or 'quote' parameter is equivalent to our stablecoin of interest.

In [5]:
```
spot_markets = client.catalog_markets(
    market_type='spot',
).to_dataframe()
```

In [6]:
```
spot_markets[['market','min_time','max_time','exchange','base','quote','symbol']]
```

Out[6]:

	market	min_time	max_time	exchange	base	quote	symbol
0	bibox-1inch-usdt-spot	2022-03-07 19:43:21.195000+00:00	2023-05-17 14:20:42.900000+00:00	bibox	1inch	usdt	1INCH_USDT
1	bibox-aaa-usdt-spot	2021-12-01 11:11:20.656000+00:00	2022-03-01 18:14:06.507000+00:00	bibox	aaa	usdt	AAA_USDT
2	bibox-aave-btc-spot	2022-03-07 19:47:10.014000+00:00	2023-05-17 14:20:41.987000+00:00	bibox	aave	btc	AAVE_BTC
3	bibox-aave-eth-spot	2022-03-07 19:47:04.545000+00:00	2023-05-17 14:20:41.987000+00:00	bibox	aave	eth	AAVE_ETH
4	bibox-aave-usdt-spot	2022-03-07 19:54:22.276000+00:00	2023-05-17 14:20:41.987000+00:00	bibox	aave	usdt	AAVE_USDT
...	...	...	...	...	...	...	...
28665	zb.com-zb-usdt-spot	2019-03-15 10:34:32+00:00	2023-05-17 12:02:41+00:00	zb.com	zb	usdt	zb_usdt
28666	zb.com-zkn-usdt-spot	2022-02-12 17:07:02+00:00	2022-05-19 11:36:39+00:00	zb.com	zkn	usdt	zkn_usdt
28667	zb.com-zrx-btc-spot	2019-03-13 23:50:03+00:00	2020-05-10 13:43:09+00:00	zb.com	zrx	btc	zrx_btc
28668	zb.com-zrx-qc-spot	2019-03-15 06:25:14+00:00	2021-01-25 06:46:56+00:00	zb.com	zrx	qc	zrx_qc
28669	zb.com-zrx-usdt-spot	2019-03-10 06:32:00+00:00	2020-05-11 06:20:28+00:00	zb.com	zrx	usdt	zrx_usdt

28670 rows × 7 columns

In [7]:
```
ticker = 'usdt'
stablecoin_markets = spot_markets.loc[(spot_markets['base']==ticker) | (spot_markets['quote']==ticker)]
```

In [8]:
```
stablecoin_markets[['market','min_time','max_time','exchange','base','quote','symbol']]
```

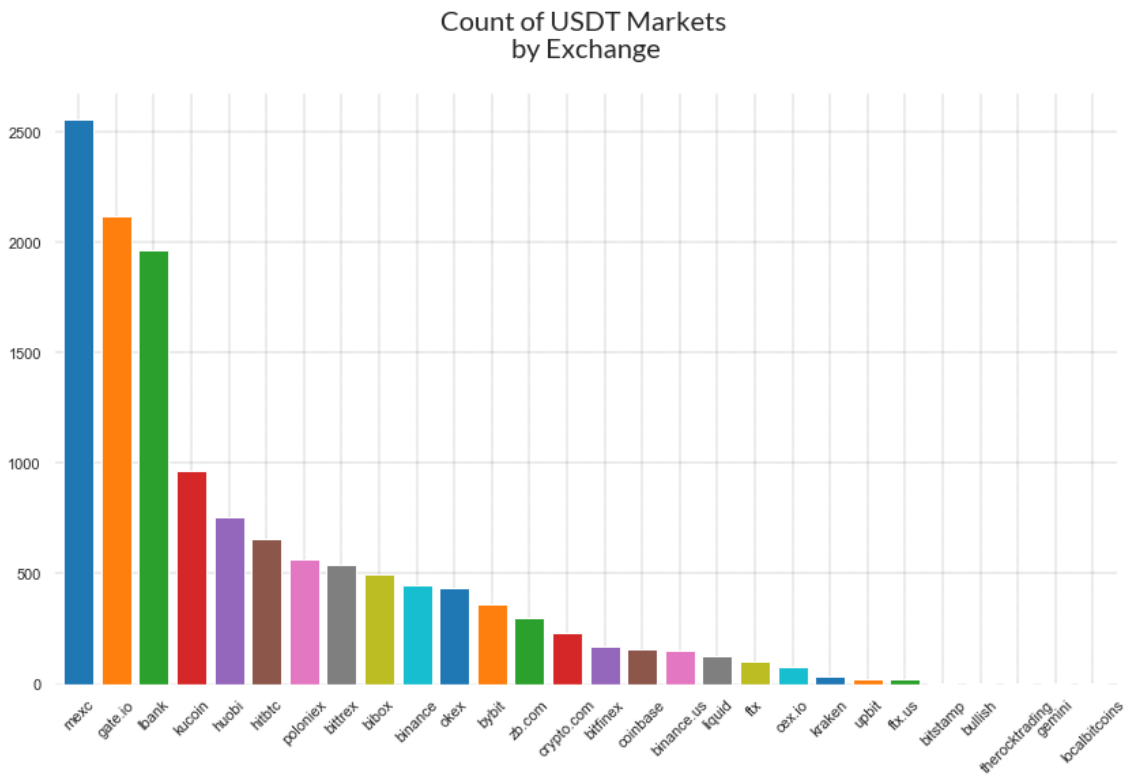
Out[8]:

	market	min_time	max_time	exchange	base	quote	symbol
0	bibox-1inch-usdt-spot	2022-03-07 19:43:21.195000+00:00	2023-05-17 14:20:42.900000+00:00	bibox	1inch	usdt	1INCH_USDT
1	bibox-aaa-usdt-spot	2021-12-01 11:11:20.656000+00:00	2022-03-01 18:14:06.507000+00:00	bibox	aaa	usdt	AAA_USDT
4	bibox-aave-usdt-spot	2022-03-07 19:54:22.276000+00:00	2023-05-17 14:20:41.987000+00:00	bibox	aave	usdt	AAVE_USDT
7	bibox-ac-usdt-spot	2022-04-02 09:34:18.007000+00:00	2023-05-06 09:41:19.011000+00:00	bibox	ac	usdt	AC_USDT
8	bibox-acmd-usdt-spot	2022-03-05 13:05:20.653000+00:00	2022-03-07 21:38:39.860000+00:00	bibox	acmd	usdt	ACMD_USDT
...	...	...	...	...	...	...	...
28660	zb.com-yfii-usdt-spot	2020-09-09 11:33:55+00:00	2021-01-28 03:10:57+00:00	zb.com	yfii	usdt	yfii_usdt
28662	zb.com-ygg-usdt-spot	2021-09-02 14:19:53+00:00	2023-04-02 09:36:42+00:00	zb.com	ygg	usdt	ygg_usdt
28665	zb.com-zb-usdt-spot	2019-03-15 10:34:32+00:00	2023-05-17 12:02:41+00:00	zb.com	zb	usdt	zb_usdt
28666	zb.com-zkn-usdt-spot	2022-02-12 17:07:02+00:00	2022-05-19 11:36:39+00:00	zb.com	zkn	usdt	zkn_usdt
28669	zb.com-zrx-usdt-spot	2019-03-10 06:32:00+00:00	2020-05-11 06:20:28+00:00	zb.com	zrx	usdt	zrx_usdt

13248 rows × 7 columns

In [9]:
```
markets_by_exchange = pd.DataFrame(stablecoin_markets['exchange'].value_counts())
```

In [10]:
```
ax1 = plt.subplot()
markets_by_exchange['exchange'].plot(kind='bar', width=0.8,color=sns.color_palette('tab10'))
plt.setp(ax1.get_xticklabels(), rotation=45);
ax1.set_facecolor("white")
plt.grid(color = 'black', linestyle = '--', linewidth = 0.2)
plt.title('\nCount of '+ str(ticker).upper() + ' Markets \nby Exchange\n',fontdict={'fontsize':20,'font':'Lato'});
```



Get stablecoin prices

Single market trades

Trades are one of the foundational data types we collect from exchanges. From raw trades data, we can construct additional aggregated metrics.

In [11]:

```
market = 'coinbase-usdt-usd-spot'
coinbase_trades = client.get_market_trades(
    markets = market,
    limit_per_market = 100,
    paging_from = 'end'
).to_dataframe()
coinbase_trades
```

Out[11]:

	market	time	coin_metrics_id	amount	price	database_time	side
0	coinbase-usdt-usd-spot	2023-05-17 14:21:41.429509+00:00	45537623	951.96	1.00006	2023-05-17 14:21:41.582837+00:00	buy
1	coinbase-usdt-usd-spot	2023-05-17 14:21:44.595979+00:00	45537624	38.92	1.00006	2023-05-17 14:21:44.834093+00:00	buy
2	coinbase-usdt-usd-spot	2023-05-17 14:21:45.196628+00:00	45537625	99.16	1.00006	2023-05-17 14:21:45.338639+00:00	buy
3	coinbase-usdt-usd-spot	2023-05-17 14:21:46.294785+00:00	45537626	0.92	1.00006	2023-05-17 14:21:46.339462+00:00	buy
4	coinbase-usdt-usd-spot	2023-05-17 14:21:49.052445+00:00	45537627	3521.95	1.00006	2023-05-17 14:21:49.091071+00:00	buy
...	...	...	...	...	...	...	...
95	coinbase-usdt-usd-spot	2023-05-17 14:23:54.487684+00:00	45537718	97.18	1.00004	2023-05-17 14:23:54.654454+00:00	sell
96	coinbase-usdt-usd-spot	2023-05-17 14:23:57.060831+00:00	45537719	8.76	1.00005	2023-05-17 14:23:57.155890+00:00	buy
97	coinbase-usdt-usd-spot	2023-05-17 14:23:57.122946+00:00	45537720	2.49	1.00004	2023-05-17 14:23:57.238913+00:00	sell
98	coinbase-usdt-usd-spot	2023-05-17 14:23:59.258734+00:00	45537721	324.76	1.00004	2023-05-17 14:23:59.406884+00:00	sell
99	coinbase-usdt-usd-spot	2023-05-17 14:24:01.240677+00:00	45537722	10.88	1.00004	2023-05-17 14:24:01.407747+00:00	sell

100 rows × 7 columns

Single market candles

From raw trades data, we construct OHLC candles for each market. Candles include the following data types:

- **price_open:** The opening price of the candle.
- **price_high:** The high price of the candle.
- **price_low:** The low price of the candle.
- **price_close:** The close price of the candle.

- **vwap:** The volume-weighted average price of the candle.
- **volume:** The volume of the candle in units of the base asset.
- **candle_usd_volume:** The volume of the candle in units of U.S. dollars.
- **candle_trades_count:** The number of trades in the candle interval.

In [12]:
```
market = 'coinbase-usdt-usd-spot'
coinbase_candles = client.get_market_candles(
    markets = market,
    frequency = '1d'
).to_dataframe()
```

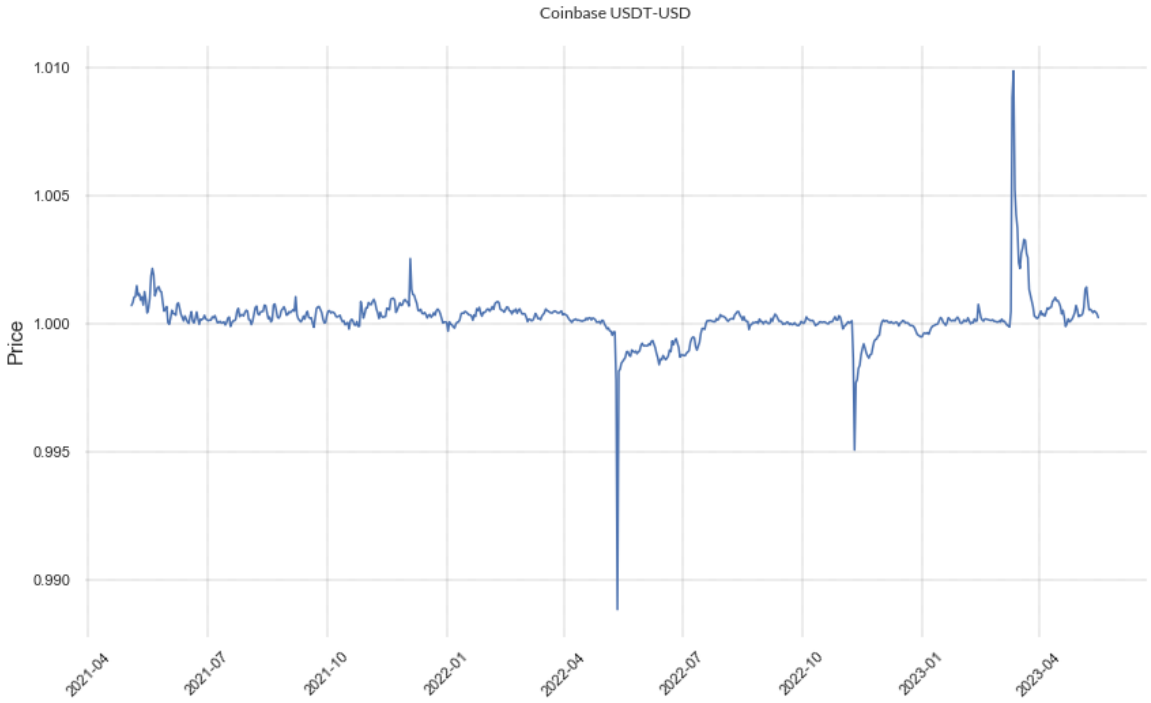
In [13]:
```
coinbase_candles
```

Out[13]:

	market	time	price_open	price_close	price_high	price_low	vwap	volume	candle_usd_volume	candle_trades_count
0	coinbase-usdt-usd-spot	2021-05-04 00:00:00+00:00	1.002	1.0006	1.003	0.999	1.000696	24564061.73	24581147.409593	30527
1	coinbase-usdt-usd-spot	2021-05-05 00:00:00+00:00	1.0006	1.0013	1.002	0.9997	1.000816	40170830.16	40203590.541009	43688
2	coinbase-usdt-usd-spot	2021-05-06 00:00:00+00:00	1.0013	1.0009	1.002	1.0004	1.001023	51129166.79	51181449.603235	51177
3	coinbase-usdt-usd-spot	2021-05-07 00:00:00+00:00	1.0008	1.0011	1.0018	1.0	1.001045	44247619.59	44293836.840656	48729
4	coinbase-usdt-usd-spot	2021-05-08 00:00:00+00:00	1.001	1.0016	1.0022	1.0009	1.00147	26972741.16	27012382.918589	50140
...	...	...	...	...	...	...	...	...	...	...
738	coinbase-usdt-usd-spot	2023-05-12 00:00:00+00:00	1.00042	1.00049	1.00067	1.00022	1.000413	96688296.14	96728224.943451	108028
739	coinbase-usdt-usd-spot	2023-05-13 00:00:00+00:00	1.00049	1.00049	1.00056	1.00044	1.000486	23602734.21	23614209.619339	42884
740	coinbase-usdt-usd-spot	2023-05-14 00:00:00+00:00	1.00049	1.00047	1.0005	1.00035	1.000441	22024368.5	22034075.523233	44754
741	coinbase-usdt-usd-spot	2023-05-15 00:00:00+00:00	1.00047	1.00024	1.00061	1.0001	1.000381	99456550.6	99494446.826319	127213
742	coinbase-usdt-usd-spot	2023-05-16 00:00:00+00:00	1.00023	1.00013	1.0015	1.00008	1.000229	89478243.93	89498691.185785	94701

743 rows × 10 columns

In [14]:
```
ax1 = plt.subplot()
coinbase_price = sns.lineplot(data=coinbase_candles,y=coinbase_candles.vwap,x=coinbase_candles.time)
plt.setp(ax1.get_xticklabels(), rotation=45);
ax1.set_facecolor("white")
plt.grid(color = 'black', linestyle = '--', linewidth = 0.2)
coinbase_price.set_xlabel("", fontsize = 15)
coinbase_price.set_ylabel("Price", fontsize = 15)
coinbase_price.set_title('\nCoinbase USDT-USD\n', fontsize = 18, font = 'Lato');
```



Reference Rate Candles

We offer reference rates quoted in USD, Euro, Bitcoin, and Ethereum. We now support these quote currencies for our entire reference rates coverage universe of over 500 assets and for all of our frequencies, including 1s, 1m, 1h, 1d-ny-close and 1d.

Current composition of markets for USDT-USD Reference Rate pair (as of May 17, 2023):

- "coinbase-usdt-usd-spot",
- "coinbase-eth-usdt-spot",
- "coinbase-btc-usdt-spot",
- "kraken-usdt-usd-spot",
- "binance-btc-usdt-spot",
- "binance-eth-usdt-spot",
- "crypto.com-usdt-usd-spot"

In [15]:
```
pairs = client.catalog_asset_pair_candles().to_dataframe()
```

In [16]:
```
pairs.loc[pairs['pair']=='usdt-usd']
```

Out[16]:

	pair	frequency	min_time	max_time
5072	usdt-usd	1m	2013-12-28 00:00:00+00:00	2023-05-17 14:22:00+00:00
5073	usdt-usd	5m	2013-12-28 00:00:00+00:00	2023-05-17 14:15:00+00:00
5074	usdt-usd	10m	2013-12-28 00:00:00+00:00	2023-05-17 14:10:00+00:00
5075	usdt-usd	15m	2013-12-28 00:00:00+00:00	2023-05-17 14:00:00+00:00
5076	usdt-usd	30m	2013-12-28 00:00:00+00:00	2023-05-17 13:30:00+00:00
5077	usdt-usd	1h	2013-12-28 00:00:00+00:00	2023-05-17 13:00:00+00:00
5078	usdt-usd	4h	2013-12-28 00:00:00+00:00	2023-05-17 08:00:00+00:00
5079	usdt-usd	1d	2013-12-28 00:00:00+00:00	2023-05-16 00:00:00+00:00

In [17]:
```
pair_candles = client.get_pair_candles(
    pairs='usdt-usd',
    start_time =  datetime.now() - timedelta(weeks=4),
    frequency='1d'
).to_dataframe()
```

In [18]:
```
pair_candles.tail()
```

Out[18]:

	pair	time	price_open	price_close	price_high	price_low
22	usdt-usd	2023-05-12 00:00:00+00:00	1.00043	1.00049	1.00067	1.00008

	pair	time	price_open	price_close	price_high	price_low
23	usdt-usd	2023-05-13 00:00:00+00:00	1.00049	1.0005	1.00056	1.00037
24	usdt-usd	2023-05-14 00:00:00+00:00	1.0005	1.00047	1.00055	1.0001
25	usdt-usd	2023-05-15 00:00:00+00:00	1.00047	1.00024	1.00058	1.00007
26	usdt-usd	2023-05-16 00:00:00+00:00	1.00024	1.00013	1.00056	0.99988

In [19]:
```
prices = pair_candles[['price_open','price_close','price_high','price_low','time']].set_index('time')
```

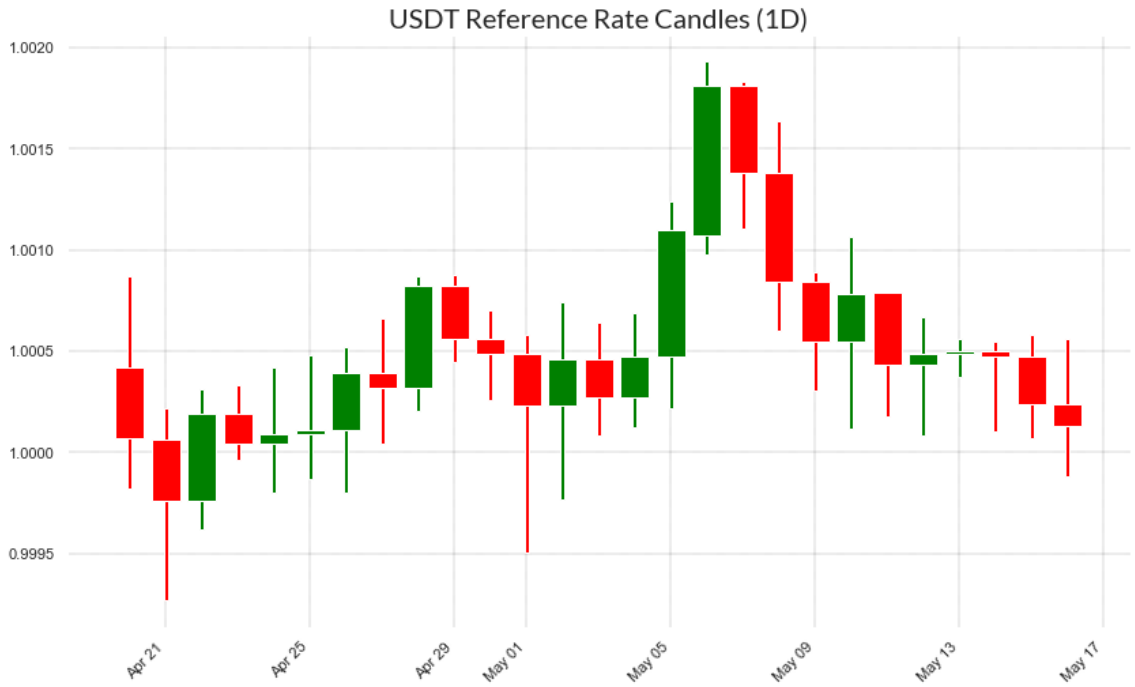
In [20]:
```
fig, ax = plt.subplots()
ax.ticklabel_format(useOffset=False)
width = 0.8
width2 = .1
ax.set_facecolor("white")
plt.grid(color = 'black', linestyle = '--', linewidth = 0.2)
plt.title('\n'+ str(ticker).upper() + ' Reference Rate Candles (1D)',fontdict={'fontsize':20,'font':'Lato'});

up = prices[prices.price_close>=prices.price_open]
down = prices[prices.price_close<prices.price_open]
col1 = 'green'
col2 = 'red'

#plot prices
ax.bar(up.index,up.price_close-up.price_open,width,bottom=up.price_open,color=col1)
ax.bar(up.index,up.price_high-up.price_close,width2,bottom=up.price_close,color=col1)
ax.bar(up.index,up.price_low-up.price_open,width2,bottom=up.price_open,color=col1)
ax.bar(down.index,down.price_close-down.price_open,width,bottom=down.price_open,color=col2)
ax.bar(down.index,down.price_high-down.price_open,width2,bottom=down.price_open,color=col2)
ax.bar(down.index,down.price_low-down.price_close,width2,bottom=down.price_close,color=col2)

#rotate x-axis tick labels
plt.xticks(rotation=45, ha='right')
ax.xaxis.set_minor_locator(MonthLocator(bymonthday=30))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %d'))

plt.show()
```



Reference Rate

In [28]:
```
rr_catalog = client.catalog_assets().to_dataframe(secondary_level='metrics')
```

In [29]:
```
rr_catalog = rr_catalog.loc[rr_catalog['asset']=='usdt']
rr_catalog = rr_catalog.loc[rr_catalog['metric']=='ReferenceRateUSD']
```

In [30]:
```
rr_catalog[['asset','full_name','metric','frequency','min_time','max_time']]
```

Out[30]:

	asset	full_name	metric	frequency	min_time	max_time
45122	usdt	Tether	ReferenceRateUSD	1d	2013-12-29 00:00:00+00:00	2023-05-17 00:00:00+00:00
45123	usdt	Tether	ReferenceRateUSD	1d-ny-close	2013-12-28 21:00:00+00:00	2023-05-16 20:00:00+00:00
45124	usdt	Tether	ReferenceRateUSD	1h	2013-12-28 01:00:00+00:00	2023-05-17 14:00:00+00:00
45125	usdt	Tether	ReferenceRateUSD	1m	2013-12-28 00:01:00+00:00	2023-05-17 14:26:00+00:00
45126	usdt	Tether	ReferenceRateUSD	1s	2013-12-28 00:00:07+00:00	2023-05-17 14:26:53+00:00

In [31]:

```
asset_rr = client.get_asset_metrics(
    assets = 'usdt',
    metrics = 'ReferenceRateUSD',
    frequency = '1m',
    start_time =  datetime.now() - timedelta(hours=24)
).to_dataframe()
```

In [32]:

```
asset_rr
```

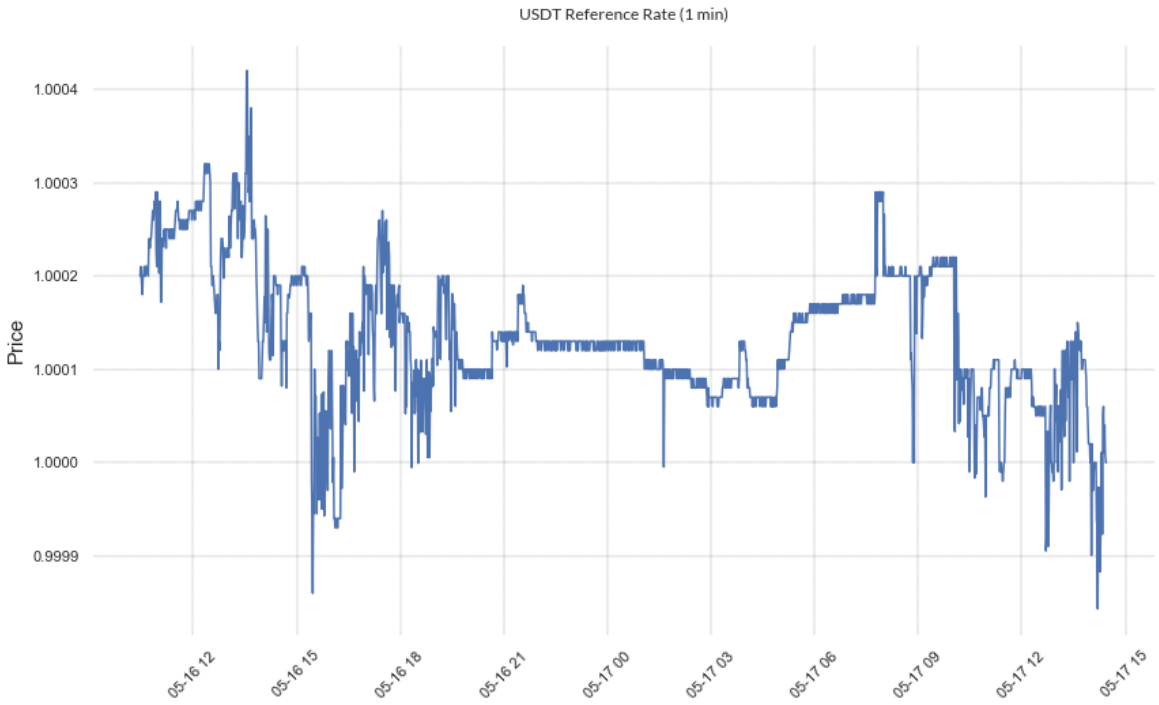
Out[32]:

	asset	time	ReferenceRateUSD
0	usdt	2023-05-16 10:28:00+00:00	1.0002
1	usdt	2023-05-16 10:29:00+00:00	1.00021
2	usdt	2023-05-16 10:30:00+00:00	1.0002
3	usdt	2023-05-16 10:31:00+00:00	1.00019
4	usdt	2023-05-16 10:32:00+00:00	1.00018
...	...	...	...
1675	usdt	2023-05-17 14:23:00+00:00	1.000011
1676	usdt	2023-05-17 14:24:00+00:00	1.00004
1677	usdt	2023-05-17 14:25:00+00:00	1.00001
1678	usdt	2023-05-17 14:26:00+00:00	1.0
1679	usdt	2023-05-17 14:27:00+00:00	1.0

1680 rows × 3 columns

In [33]:

```
ax1 = plt.subplot()
ax1.ticklabel_format(useOffset=False)
asset_rr_chart = sns.lineplot(data=asset_rr,y=asset_rr.ReferenceRateUSD,x=asset_rr.time)
plt.setp(ax1.get_xticklabels());
ax1.set_facecolor("white")
plt.grid(color = 'black', linestyle = '--', linewidth = 0.2)
asset_rr_chart.set_xlabel("", fontsize = 15)
asset_rr_chart.set_ylabel("Price", fontsize = 15)
asset_rr_chart.set_title('\nUSDT Reference Rate (1 min)\n', fontsize = 18, font = 'Lato');
```



Principal Market Price

The Principal Market Prices identify a principal market for each asset and utilize the most recent price from this market. Common use cases are for fair value measurement, preparing financial statements, and calculating closing prices for indexes or financial benchmarks.

In [56]:
```
asset_pmp = client.get_asset_metrics(
    assets = 'usdt',
    metrics = ['principal_market_price_usd','principal_market_usd'],
    frequency = '1m',
    start_time =  datetime.now() - timedelta(hours=24)
).to_dataframe()
```

In [57]:
```
asset_pmp
```

Out[57]:

	asset	time	principal_market_price_usd	principal_market_usd
0	usdt	2023-05-16 10:45:00+00:00	1.00023	coinbase-usdt-usd-spot
1	usdt	2023-05-16 10:46:00+00:00	1.00024	coinbase-usdt-usd-spot
2	usdt	2023-05-16 10:47:00+00:00	1.00024	coinbase-usdt-usd-spot
3	usdt	2023-05-16 10:48:00+00:00	1.00025	coinbase-usdt-usd-spot
4	usdt	2023-05-16 10:49:00+00:00	1.00026	coinbase-usdt-usd-spot
...	...	...	...	...
1675	usdt	2023-05-17 14:40:00+00:00	1.00001	coinbase-usdt-usd-spot
1676	usdt	2023-05-17 14:41:00+00:00	1.0	coinbase-usdt-usd-spot
1677	usdt	2023-05-17 14:42:00+00:00	1.0	coinbase-usdt-usd-spot
1678	usdt	2023-05-17 14:43:00+00:00	1.00001	coinbase-usdt-usd-spot
1679	usdt	2023-05-17 14:44:00+00:00	1.00001	coinbase-usdt-usd-spot

1680 rows × 4 columns

In [58]:
```
market_list = list(set(asset_pmp['principal_market_usd'].to_list()))
market_list
```

Out[58]:
```
['coinbase-usdt-usd-spot', 'kraken-usdt-usd-spot']
```

In [124…
```
unique_markets = asset_pmp['principal_market_usd'].unique()
colors = plt.cm.jet(np.linspace(0,1,len(unique_markets)))
color_map = dict(zip(unique_markets, colors))

asset_pmp = asset_pmp.sort_values('time')
fig, ax = plt.subplots()
```

```
ax.set_facecolor("white")
ax.grid(color='lightgray', linestyle='--')

for i in range(1, len(asset_pmp)):
    ax.plot(asset_pmp['time'].iloc[i-1:i+1],
            asset_pmp['principal_market_price_usd'].iloc[i-1:i+1],
            color=color_map[asset_pmp['principal_market_usd'].iloc[i]],
            label=asset_pmp['principal_market_usd'].iloc[i])

handles, labels = plt.gca().get_legend_handles_labels()
by_label = dict(zip(labels, handles))
import matplotlib.ticker as ticker

ax.legend(by_label.values(), by_label.keys(), loc='upper right', frameon=False, bbox_to_anchor=(1, 1.135))
ax.get_yaxis().set_major_formatter(ticker.FuncFormatter(lambda x, p: format(float(x), '.4f')))

plt.title('USDT Principal Market Price\n', fontsize=22)
plt.xlabel('')
plt.ylabel('Principal Market Price (USD)\n',fontsize=16)
plt.show()
```