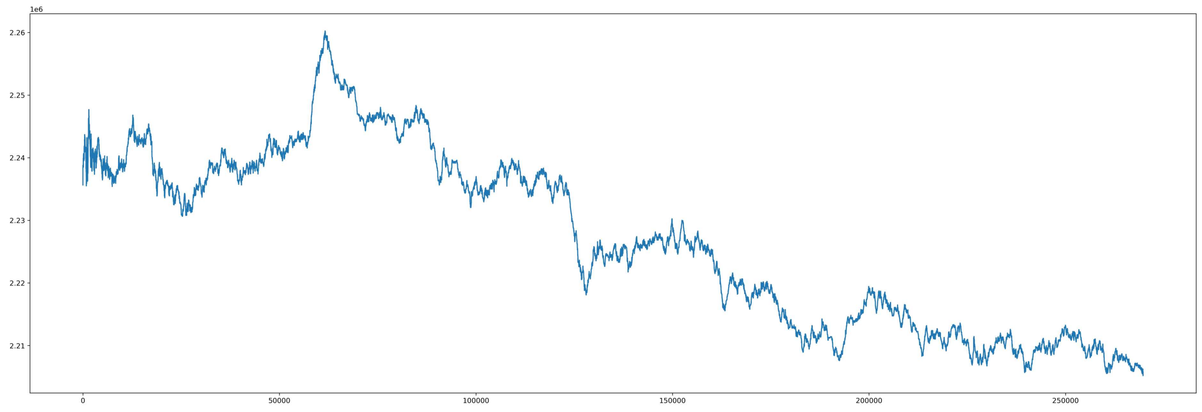
mid price 概览

In [18]:

```
import numpy as np
import pandas as pd
import seaborn as sn
from matplotlib import pyplot as plt
df = pd.read_csv('order.csv')
ask_price = df['ask_price']
bid_price = df['bid_price']
mid_price = (ask_price + bid_price)/2
df['mid'] = mid_price
plt.figure(dpi = 300, figsize = (30,10))
sn.lineplot(data=mid_price)
plt.show()
```



In [19]:

```
print(mid_price)
```

```
0         2235650.0
1         2238800.0
2         2238800.0
3         2238800.0
4         2238800.0
            ...
269743    2205650.0
269744    2205750.0
269745    2205750.0
269746    2205700.0
269747    2205750.0
Length: 269748, dtype: float64
```

In [20]:

```
df
```

Out[20]:

	ask_price	ask_size	bid_price	bid_size	mid
0	2239500	100	2231800	100	2235650.0
1	2239500	100	2238100	21	2238800.0
2	2239500	100	2238100	21	2238800.0
3	2239500	100	2238100	21	2238800.0
4	2239500	100	2238100	21	2238800.0
...	...	...	...	...	...
269743	2206200	100	2205100	249	2205650.0
269744	2206400	100	2205100	249	2205750.0
269745	2206400	100	2205100	249	2205750.0
269746	2206300	100	2205100	249	2205700.0
269747	2206400	100	2205100	249	2205750.0

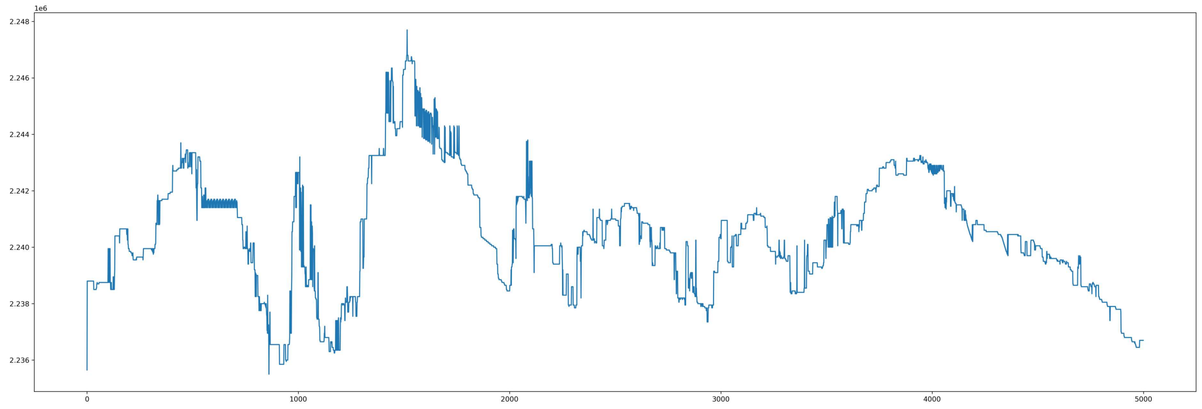
269748 rows × 5 columns

midprice在5000个单位上的缩放图

In [22]:

```python
min_mid = mid_price[:5000]
print(min_mid)
plt.figure(dpi = 300, figsize = (30,10))
sn.lineplot(data=min_mid)
plt.show()
```

```
0       2235650.0
1       2238800.0
2       2238800.0
3       2238800.0
4       2238800.0
          ...
4995    2236700.0
4996    2236700.0
4997    2236700.0
4998    2236700.0
4999    2236700.0
Length: 5000, dtype: float64
```

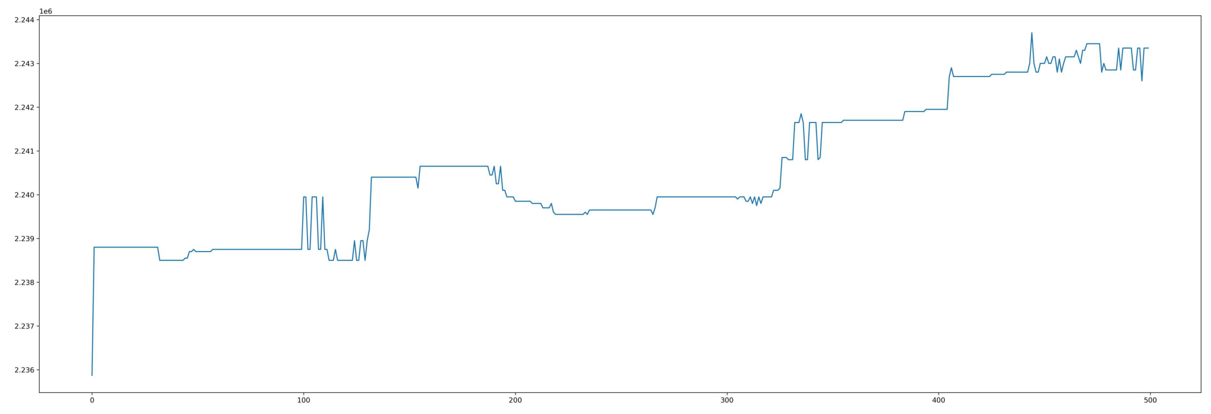


假设midprice作为一个变量x，x仅被时序和x本身所解释（X单纯拥有markov性质）。且x在时间T上仅仅与[X_{T-500}，X_T]有关（这里简化我们的分析过程，让num：n仅仅为500）
建立线性回归模型， X=[1,500]为时序，y=[前一百个mid_price] 然后预测T+1上的值。根据当前价和预测值进行多空操作，统计后m个序列中是否会盈利（m暂定50）

In [62]:
```
def filter_extreme_3sigma(data,n=3,times=3):
    # times进行times次3sigma处理 去极值
    series = data.copy()
    for i in range(times):
        mean = series.mean()
        std = series.std()
        max_range = mean + n*std
        min_range = mean - n*std
        series = np.clip(series,min_range,max_range)
    return series

from sklearn.linear_model import LinearRegression

X = np.arange(0,500,1).reshape(-1, 1)
y = mid_price[:500]
y = filter_extreme_3sigma(y)
plt.figure(dpi = 300, figsize = (30,10))
sn.lineplot(data=y)
plt.show()
print(y)
reg = LinearRegression().fit(X, y)
print("current price ", mid_price[500]) # 当前
print("predict price ", reg.predict([[550]]))
```



```
0      2.235872e+06
1      2.238800e+06
2      2.238800e+06
3      2.238800e+06
4      2.238800e+06
           ...
495    2.243350e+06
496    2.242600e+06
497    2.243350e+06
498    2.243350e+06
499    2.243350e+06
Length: 500, dtype: float64
current price  2243350.0
predict price  [2243429.26127444]
```

In [61]:
```
wait = mid_price[500:550]
wait
```

Out[61]:
```
500    2243350.0
501    2243350.0
502    2243350.0
503    2243350.0
504    2243350.0
505    2243350.0
506    2243350.0
507    2243350.0
508    2243350.0
509    2243350.0
510    2243350.0
511    2243350.0
512    2243350.0
513    2243350.0
514    2243350.0
515    2243000.0
516    2242100.0
517    2242100.0
518    2242750.0
519    2241600.0
520    2240950.0
521    2240950.0
522    2242100.0
523    2242250.0
524    2242250.0
525    2243200.0
526    2243200.0
527    2243200.0
528    2243200.0
529    2243200.0
530    2243200.0
531    2243200.0
532    2243200.0
533    2243200.0
534    2243050.0
535    2243050.0
536    2243050.0
537    2243050.0
538    2243050.0
539    2242100.0
540    2242100.0
541    2242100.0
542    2242100.0
543    2241400.0
544    2241800.0
545    2241800.0
546    2241800.0
547    2241400.0
548    2241650.0
549    2241650.0
dtype: float64
```

2243429.26127444 高于 2239950.0值（当前值）按照预测结果则会做多 从后面的50个mid_price来看这一次交易是有效的
为了进一步分析可行性，我们统计随机tick上一千次尝试中盈亏（实际上在一千次里做回测）止盈定在预测值附近200以内 止损定在50个tick上

In [63]:

```
am = 0 # 总盈利
for i in range(1000):
    X = np.arange(0,500,1).reshape(-1, 1)
    y = mid_price[:500]
    y = filter_extreme_3sigma(y)
# plt.figure(dpi = 300, figsize = (30,10))
# sn.lineplot(data=y)
# plt.show()
# print(y)
    reg = LinearRegression().fit(X, y)
    predit = reg.predict([[500]])
    current = mid_price[i+500]
    # 计算return(包括止盈止损)
    wait = mid_price[i+500:i+550].tolist()
    flag = predit - current
    ret = 0
    if flag > 0:
        for x in wait:
            if x > predit - 200:
                ret = x - current
                print("做多： ", ret)
                break
        if ret == 0:
            ret =wait[-1] - current
            print("做多： ", ret)
    if flag < 0:
        for x in wait:
            if x < predit + 200:
                ret = current - x
                print("做空： ", ret)
                break
        if ret == 0:
            ret =  current - wait[-1]
            print("做空： ", ret)

    if flag == 0:
        continue
    am = am + ret
    print("总盈利",am)


am

# print(reg.predict([[100]]))
# print(mid_price[101])
```

```
做空：  -2200.0
总盈利 120900.0
做空：  -2200.0
总盈利 118700.0
做空：  -2050.0
总盈利 116650.0
做空：  -2150.0
总盈利 114500.0
做空：  -2150.0
总盈利 112350.0
做空：  -2350.0
```

```
总盈利 110000.0
做空： -500.0
总盈利 109500.0
做空： -500.0
总盈利 109000.0
做空： -500.0
总盈利 108500.0
做空： -500.0
总盈利 108000.0
```

下面试试一万个tick上的预测盈亏

In [64]:

```
am = 0 # 总盈利
for i in range(10000):
    X = np.arange(0,500,1).reshape(-1, 1)
    y = mid_price[:500]
    y = filter_extreme_3sigma(y)
    reg = LinearRegression().fit(X, y)
    predit = reg.predict([[500]])
    current = mid_price[i+500]
    # 计算return(包括止盈止损)
    wait = mid_price[i+500:i+550].tolist()
    flag = predit - current
    ret = 0
    if flag > 0:
        for x in wait:
            if x > predit - 200:
                ret = x - current
                print("做多: ", ret)
                break
        if ret == 0:
            ret =wait[-1] - current
            print("做多: ", ret)
    if flag < 0:
        for x in wait:
            if x < predit + 200:
                ret = current - x
                print("做空: ", ret)
                break
        if ret == 0:
            ret =  current - wait[-1]
            print("做空: ", ret)

    if flag == 0:
        continue
    am = am + ret
    print("总盈利",am)


am
```

```
做多:  -200.0
总盈利 413750.0

做多:  -200.0
总盈利 413550.0
做多:  -250.0
总盈利 413300.0
做多:  -250.0
总盈利 413050.0
做多:  -250.0
总盈利 412800.0
做多:  -250.0
总盈利 412550.0
做多:  -250.0
总盈利 412300.0
做多:  -300.0
```

```
总盈利 412000.0
做多： -300.0
总盈利 411700.0
做多： -300.0
```

一万次的试验下达成409900的总盈利。以上已经能证明可行性。

改进提议：
1 根据小止损大止盈原则，可以加上一个跳价shift_price
2 数据中很多时刻重复值 不能每一个tick都开仓 可以间隔n个tick开一次
3 m和n参数化，并针对参数进行调优，优化损失曲线
4 假设增加，除了Markov性质，建立基于当前时刻订单簿和模型预测值具有相关性的假设。例如判断k档订单部深度，选择深度大的一边，如和模型多空达成一致性则作为有效信号开仓
5 模型中对r2进行判断，基于r2进行信号筛选。r2大于threshold才作为有效信号开仓