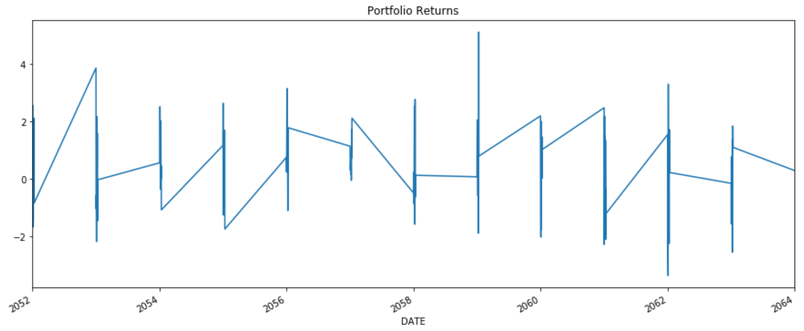
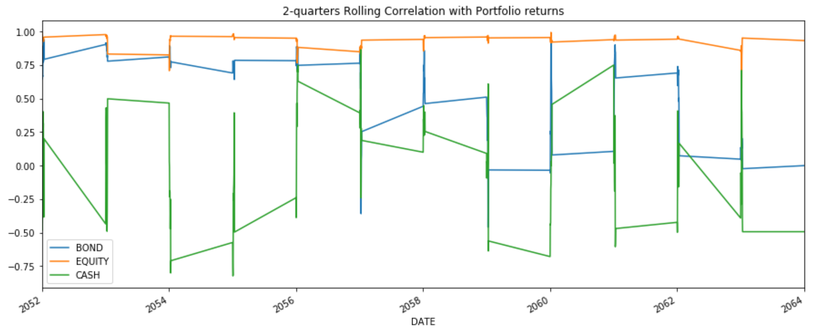
In [92]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.stats.stattools import durbin_watson
from sklearn.linear_model import LinearRegression, LassoLars, lars_path, lasso_path
from sklearn.preprocessing import normalize

prices = pd.read_csv('dataAL.csv', index_col=0, parse_dates=True).dropna()
prices.drop(prices.index[0])
trainingdates = np.intersect1d(prices.index, pd.date_range('2052-01-01','2064-01-01'))
cvdates = np.intersect1d(prices.index, pd.date_range('2060-01-01','2064-01-01'))

areturns=prices.pct_change()[1:]
weights=[30,30,40]
preturns=areturns.dot(weights)
preturns = preturns.rename('preturns')
preturns[trainingdates].plot(figsize=(15,6), title='Portfolio Returns')
plt.show()
```



In [93]:
```
#Correlation analysis for predictors
correls = pd.concat([preturns, areturns], axis=1).rolling(window=4*2).corr()
correls
correls = correls.loc[(slice(None), 'preturns'),:].dropna().drop('preturns', axis=1)
correls.index = correls.index.droplevel(level=1)

correls.loc[trainingdates].plot(figsize=(15,6), title='2-quarters Rolling Correlation with Portfolio returns')
plt.show()
```



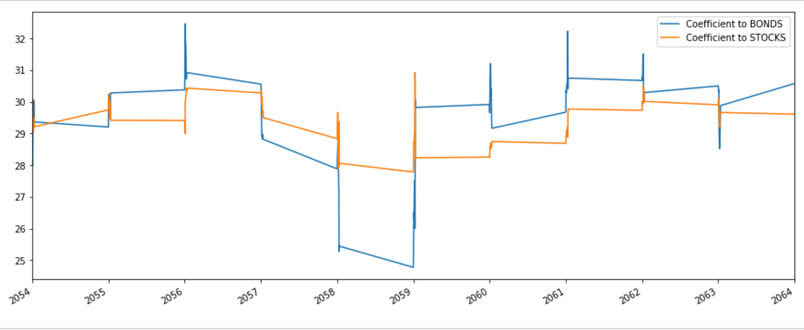
In [94]:
```
#Select less correlated predictors
X=areturns.loc[trainingdates].values
Y=preturns.loc[trainingdates].values
selected_predictors = areturns[['BOND', 'EQUITY']]

np.corrcoef(X, rowvar=False)
```

Out[94]:
```
array([[ 1.        ,  0.13757227, -0.0715795 ],
       [ 0.13757227,  1.        ,  0.09785607],
       [-0.0715795 ,  0.09785607,  1.        ]])
```

In [95]:
```
def regress(predictor, response, start, end, halflife, returnr2=False, outlier_cap=2):
    '''
        helper function to run regression
        predictor: dataframe of predictors
        response: series of response
        start: start date
        end: end date
        halflife: halflife to weight regression exponentially
    '''
    y = response[start:end].values
    X = predictor[start:end].values
    # cap both y and X at 2 standard deviations to avoid heavy influence of outliers
    y = np.clip(y, -outlier_cap * np.std(y), outlier_cap * np.std(y))
    X = np.clip(X, -outlier_cap * np.std(X, axis=0), outlier_cap * np.std(X, axis=0))

    assert X.shape[0] == y.shape[0], "predictor and response are not the same length"

    w = np.ones(y.shape)
    decay = 2.0 ** (-1.0/halflife)
    for i in range(len(w)-2, -1, -1):
        w[i] = w[i+1] * decay

    reg = LinearRegression(normalize=True)
    reg.fit(X, y, sample_weight=w)
    if returnr2:
        return (reg, reg.score(X,y))
    return reg
```

In [96]:
```
# Rolling regression
regression_window = 4* 6
regression_halflife = 4 * 6
reg = np.empty(len(trainingdates), dtype=object)
for t in range(regression_window, len(trainingdates)):
    reg[t] = regress(selected_predictors, preturns,
                     trainingdates[t-regression_window], trainingdates[t],
                     regression_halflife)
regobj = pd.Series(reg[regression_window:], index=trainingdates[regression_window:], name='regress_object')
```

In [97]:
```
#Change of Regression Coefficients over time
plt.figure(figsize=(15,6))
regobj.map(lambda x: x.coef_[0]).plot(label='Coefficient to BONDS')
regobj.map(lambda x: x.coef_[1]).plot(label='Coefficient to STOCKS')

plt.legend()
plt.show()
```



In [ ]:

In [ ]:

In [ ]: