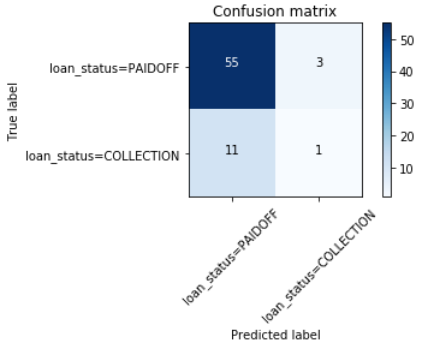
Out[171]: 0.8

In [172]:
```
from sklearn.metrics import classification_report, confusion_matrix
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
print(confusion_matrix(y_test, yhat_test2, labels=['PAIDOFF', 'COLLECTION']))
```

```
[[55  3]
 [11  1]]
```

In [173]:
```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, yhat_test2, labels=['PAIDOFF', 'COLLECTION'])
np.set_printoptions(precision=2)


# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=['loan_status=PAIDOFF','loan_status=COLLECTION'],normalize= False,  title='Confusion matr
ix')
```

```
Confusion matrix, without normalization
[[55  3]
 [11  1]]
```



In [174]:
```
print (classification_report(y_test, yhat_test2))
```

```
              precision    recall  f1-score   support

  COLLECTION       0.25      0.08      0.12        12
     PAIDOFF       0.83      0.95      0.89        58

   micro avg       0.80      0.80      0.80        70
   macro avg       0.54      0.52      0.51        70
weighted avg       0.73      0.80      0.76        70
```

Report

You should be able to report the accuracy of the built model using different evaluation metrics:

Algorithm	Jaccard	F1-score	LogLoss
KNN	2	2	NA