

5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} On this page

Testing Hooks

Testing hooks

Testing hooks is same as testing contracts. The template includes a test for the Counter hook, which you can find [intest/Counter.t.sol](#) .

Here are some key points about the Counter hook test:

1. The hook extends from a couple of utilities that facilitate easier testing of hooks.
2. import
3. "forge-std/Test.sol"
4. ;
5. import
6. {
7. Deployers
8. }
9. from
10. "v4-core/test/utis/Deployers.sol"
11. ;
12. contract
13. CounterTest
14. is
15. Test
16. ,
17. Deployers
18. {
19. }
20. Copy
21. Thesetup
22. function, called before every test, creates a few test tokens, retrieves the hook address, and then initializes the pool with this hook address.
23. function
24. setup
25. (
26.)
27. public
28. {
29. Deployers
30. .
31. deployFreshManagerAndRouters
32. (
33.)
34. ;
35. (
36. currency0
37. ,
38. currency1
39.)
40. =
41. Deployers
42. .
43. deployMintAndApprove2Currencies
44. (
45.)
46. ;
47. // Deploy the hook to an address with the correct flags
48. uint160
49. flags
50. =

```
51. uint160
52. (
53. Hooks
54. .
55. BEFORE_SWAP_FLAG
56. |
57. Hooks
58. .
59. AFTER_SWAP_FLAG
60. |
61. Hooks
62. .
63. BEFORE_ADD_LIQUIDITY_FLAG
64. |
65. Hooks
66. .
67. BEFORE_REMOVE_LIQUIDITY_FLAG
68. )
69. ;
70. (
71. address
72. hookAddress
73. ,
74. bytes32
75. salt
76. )
77. =
78. HookMiner
79. .
80. find
81. (
82. address
83. (
84. this
85. )
86. ,
87. flags
88. ,
89. type
90. (
91. Counter
92. )
93. .
94. creationCode
95. ,
96. abi
97. .
98. encode
99. (
100. address
101. (
102. manager
103. )
104. )
105. )
106. ;
107. counter
108. =
109. new
110. Counter
111. {
112. salt
113. :
114. salt
115. }
116. (
117. IPoolManager
118. (
```

```
119. address
120. (
121. manager
122. )
123. )
124. )
125. ;
126. }
127. Copy
128. Pool is then initialized containing this hook
129. // Create the pool
130. key
131. =
132. PoolKey
133. (
134. currency0
135. ,
136. currency1
137. ,
138. 3000
139. ,
140. 60
141. ,
142. IHooks
143. (
144. counter
145. )
146. )
147. ;
148. poolId
149. =
150. key
151. .
152. told
153. (
154. )
155. ;
156. initializeRouter
157. .
158. initialize
159. (
160. key
161. ,
162. SQRT_RATIO_1_1
163. ,
164. ZERO_BYTES
165. )
166. ;
167. Copy
168. Hook tests utilize a router, namely PoolModifyPositionTest
169. , to modify positions. PoolModifyPositionTest implements the LockCallback
170. interface and adds the lockAcquired
171. function, which in turn calls the manager.modifyPosition
172. function.
173. PoolManager manager
174. ;
175. PoolModifyPositionTest modifyPositionRouter
176. ;
177. manager
178. =
179. new
180. PoolManager
181. (
182. 500000
183. )
184. ;
185. // Helpers for interacting with the pool
186. modifyPositionRouter
```

```

187. =
188. new
189. PoolModifyPositionTest
190. (
191. IPoolManager
192. (
193. address
194. (
195. manager
196. )
197. )
198. )
199. ;
200. modifyPositionRouter
201. .
202. modifyPosition
203. (
204. poolKey
205. ,
206. IPoolManager
207. .
208. ModifyPositionParams
209. (
210. -
211. 120
212. ,
213. 120
214. ,
215. 10
216. ether
217. )
218. ,
219. ZERO_BYTES
220. )
221. ;
222. Copy
223. Similarly, for token swaps, the test uses PoolSwapTest
224. , which also implements the ILockCallback
225. interface.
226. Testing the hook closely resembles testing any other smart contract. The function testCounterHooks
227. executes swaps and verifies if the counters are updated correctly.
228. function
229. testCounterHooks
230. (
231. )
232. public
233. {
234. // positions were created in setup()
235. assertEq
236. (
237. counter
238. .
239. beforeAddLiquidityCount
240. (
241. poolId
242. )
243. ,
244. 3
245. )
246. ;
247. assertEq
248. (
249. counter
250. .
251. beforeRemoveLiquidityCount
252. (
253. poolId
254. )

```

```
255. ,
256. 0
257. )
258. ;
259. assertEquals
260. (
261. counter
262. .
263. beforeSwapCount
264. (
265. poolId
266. )
267. ,
268. 0
269. )
270. ;
271. assertEquals
272. (
273. counter
274. .
275. afterSwapCount
276. (
277. poolId
278. )
279. ,
280. 0
281. )
282. ;
283. // Perform a test swap //
284. bool
285. zeroForOne
286. =
287. true
288. ;
289. int256
290. amountSpecified
291. =
292. 1e18
293. ;
294. BalanceDelta swapDelta
295. =
296. swap
297. (
298. key
299. ,
300. zeroForOne
301. ,
302. amountSpecified
303. ,
304. ZERO_BYTES
305. )
306. ;
307. // ----- //
308. assertEquals
309. (
310. int256
311. (
312. swapDelta
313. .
314. amount0
315. (
316. )
317. )
318. ,
319. amountSpecified
320. )
321. ;
322. assertEquals
```

```
323. (  
324. counter  
325. .  
326. beforeSwapCount  
327. (  
328. poolId  
329. )  
330. ,  
331. 1  
332. )  
333. ;  
334. assertEq  
335. (  
336. counter  
337. .  
338. afterSwapCount  
339. (  
340. poolId  
341. )  
342. ,  
343. 1  
344. )  
345. ;  
346. }  
347. /// Test Helper  
348. function  
349. swap  
350. (  
351. PoolKey  
352. memory  
353. key  
354. ,  
355. bool  
356. zeroForOne  
357. ,  
358. int256  
359. amountSpecified  
360. ,  
361. bytes  
362. memory  
363. hookData  
364. )  
365. internal  
366. returns  
367. (  
368. BalanceDelta delta  
369. )  
370. {  
371. IPoolManager  
372. .  
373. SwapParams  
374. memory  
375. params  
376. =  
377. IPoolManager  
378. .  
379. SwapParams  
380. (  
381. {  
382. zeroForOne  
383. :  
384. zeroForOne  
385. ,  
386. amountSpecified  
387. :  
388. amountSpecified  
389. ,  
390. sqrtPriceLimitX96
```

```
391. :
392. zeroForOne
393. ?
394. TickMath
395. .
396. MIN_SQRT_RATIO
397. +
398. 1
399. :
400. TickMath
401. .
402. MAX_SQRT_RATIO
403. -
404. 1
405. // unlimited impact
406. }
407. )
408. ;
409. PoolSwapTest
410. .
411. TestSettings
412. memory
413. testSettings
414. =
415. PoolSwapTest
416. .
417. TestSettings
418. (
419. {
420. withdrawTokens
421. :
422. true
423. ,
424. settleUsingTransfer
425. :
426. true
427. ,
428. currencyAlreadySent
429. :
430. false
431. }
432. )
433. ;
434. delta
435. =
436. swapRouter
437. .
438. swap
439. (
440. key
441. ,
442. params
443. ,
444. testSettings
445. ,
446. hookData
447. )
448. ;
449. }
450. Copy Edit this page .css-1tclyyl{margin-top:1.5rem;} .css-1c3fvx8{display:-webkit-box;display:-webkit-flex;display:-ms-
flexbox;display:flex;-webkit-flex-direction:row;-ms-flex-direction:row;flex-direction:row;-webkit-align-items:center;-
webkit-box-align:center;-ms-flex-align:center;align-items:center;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-
justify-content:center;justify-content:center;} .css-1wsnqg4{font-size:1rem;padding-right:0.5rem;} Helpful? .css-
y2jwfw{fill:transparent;opacity:0.5;}.css-y2jwfw:hover{fill:#5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} Previous Building your own hook Next Hook
Deployment * Testing hooks
```

