Charles Proxy - Detailed Notes

1. Introduction

Charles Proxy is a web debugging proxy application that lets you monitor, capture, and manipulate network traffic between your computer/mobile device and the internet.

Why do testers use it?
- Debugging API requests and responses
- Testing mobile apps (Android/iOS)
- Simulating poor network conditions
- Modifying requests (Rewrite, Breakpoints)
- Checking SSL/HTTPS traffic

2. How Charles Proxy Works

Charles acts as a man-in-the-middle (MITM) between your app/browser and the server.
Steps:
1. Request goes from App → Charles → Server
2. Response comes from Server → Charles → App
3. Charles shows the full request & response details.

3. Key Features

- **SSL Proxying:** Allows you to view HTTPS traffic by installing Charles Root Certificate. Example: Viewing a login API request with sensitive data.

- **Breakpoints:** Pause requests/responses before they reach the server/app. Example: Pause a payment API request, edit amount = 1 → 1000, then forward.

In a food delivery app, change the **price value** in the request from `₹500` → `₹1` before sending it to the server.

- **Rewrite Tool:** Automatically modify requests/responses. Example: Replace ENV=Production with ENV=QA.

Rewrite all instances of `"USD"` → `"INR"` in responses to check how the app handles local currency.

- **Throttling:** Simulate slow or unstable networks. Example: Test app behavior when internet speed = 2G.

Navya Shree H

🔹 **Map Local:** Map API calls to a local JSON file instead of real server. Example: Test cart API by mapping to a fake response {“items”: 5}.

Use a **local JSON file** as a fake response for testing product catalog in an e-commerce app.

🔹 **Map Remote:** Redirect traffic from one server to another. Example: Redirect api.prod.com → api.qa.com.

Redirect staging API (`staging.api.flipkart.com`) → production API (`api.flipkart.com`) for testing without changing app config.

4. Setup

On Desktop

1. Install Charles Proxy.
2. Open browser and configure proxy = 127.0.0.1:8888.
3. Start capturing traffic.

On Mobile (Android/iOS)

1. Connect mobile to same WiFi as PC.
2. Set proxy in WiFi = PC IP & Port 8888.
3. Install Charles Root Certificate on mobile.
4. Start capturing mobile app traffic.

5. Real-Time Examples

✔ Login API Debugging:
POST /login
Body: { "username": "test", "password": "1234" }
Response: { "status": "success", "token": "abc123" }

✔ Payment Gateway Simulation:
POST /payment
Amount: 500
(Add Breakpoint → Change amount 500 → 1 → Forward)
Response: Payment success with ₹1 deducted.

✔ Testing Slow Internet:
Enable Throttling → 2G. Open shopping app → Images load slowly.

✔ Map Local – Fake Order API:
GET /orders
Response: { "orderCount": 0 }

Map Local → Fake JSON file → { "orderCount": 10 }
App shows '10 orders' without backend change.

✔ Rewrite Example:
Rule: Replace currency=USD → currency=INR
App shows prices in INR instead of USD.

6. Common Real-Time Use Cases

✔ Debugging login, signup, checkout APIs
✔ Validating error handling (404, 500 errors)
✔ Testing push notifications APIs
✔ Simulating expired tokens/session handling
✔ Checking analytics events (Google Analytics, Firebase, etc.)

7. Best Practices

- Always use Charles in test environments (not production with sensitive data).
- Turn off proxy when not needed (security).
- Save Charles logs (.chls) and share with developers for debugging.

8. Charles Proxy Alternatives

- **Fiddler** (Windows)
- **Wireshark** (More advanced, packet-level)
- **Proxyman** (Mac)

9. Conclusion

Charles Proxy is an essential tool for testers to:
- Inspect, Modify, and Debug APIs
- Simulate real-world network conditions
- Validate app behavior without depending fully on backend changes

Charles Proxy - Real-Time Interview Questions and Answers

1) What is Charles Proxy? Why do we use it in testing?

Charles Proxy is a web debugging proxy tool that captures HTTP/HTTPS traffic between your computer or mobile device and the internet. It's mainly used for:
- Analyzing API requests/responses
- Debugging network failures
- Modifying requests/responses
- Testing slow networks or error cases

2) How do you configure Charles Proxy to capture traffic from a mobile device?

1. Connect the mobile device and computer to the same Wi-Fi.
2. Note the IP address of your computer (where Charles is installed).
3. In mobile device Wi-Fi settings:
   - Set manual proxy with:
     - Host: IP of your computer
     - Port: 8888 (default Charles port)
4. Open Charles → Go to Proxy > SSL Proxying Settings → Enable SSL Proxying.
5. Install the Charles SSL certificate on the mobile browser.

3) How do you install the Charles SSL certificate on Android/iOS?

Android:
- Visit http://charlesproxy.com/getssl
- Download and install the certificate
- Enable 'Trust user certificates' in developer settings (for Android 11+).

iOS:
- Visit same link in Safari → download profile
- Go to Settings > Profile Downloaded > Install
- Trust the certificate via Settings > General > About > Certificate Trust Settings

4) How do you use breakpoints in Charles Proxy?

1. Enable breakpoints via Proxy > Breakpoints.
2. Right-click on a request → Enable breakpoint.
3. Modify request/response before sending or receiving.
4. Click Execute to proceed.

Use case: Changing parameters to test various scenarios.

5) How do you simulate a network failure or delay in Charles?

Go to Proxy > Throttle Settings → Enable Throttling.
Set options like:
- Bandwidth limit
- Latency
- DNS failure simulation

Useful to test app behavior on poor networks.

6) You see a 401 Unauthorized error in Charles. What steps will you take?

- Check Authorization headers
- Verify token validity
- Compare with successful Postman calls
- Check token expiry/refresh flow
- Confirm user permissions for the endpoint

7) How do you use Map Local and Map Remote?

Map Local: Redirects request to a local file for mocking responses.
Map Remote: Redirects request to a different URL.
Use cases: A/B testing, simulate API versions or error responses.

8) How to export and share Charles logs?

Go to File > Save Session As... and save as a .chls file.
Share the file with the developer for debugging.

9) How to check if caching is causing API issues in Charles?

- Look for Cache-Control, Expires, ETag headers
- 304 Not Modified indicates cache
- Disable cache in Charles or app/browser settings
- Use No Caching option in Charles

10) How do you identify if the issue is on frontend or backend using Charles?

- No request to server: Frontend issue
- Request hits server but error response: Backend issue
- Correct response but app breaks: Frontend handling issue

Use Charles to analyze payloads and status codes

11) How does Charles Proxy work as a MITM (Man-in-the-Middle) proxy at the SSL/TLS layer?

Charles Proxy inserts itself between the client (mobile app/browser) and the server.
- Normally, SSL/TLS communication is encrypted end-to-end.
- Charles breaks this encryption by generating its own self-signed root certificate.
- When we install this certificate on the device, the device trusts Charles as a Certificate Authority (CA).

- So, traffic is decrypted inside Charles, inspected, and then re-encrypted before reaching the real server.

👉 This is why it's called a MITM proxy.

12) If a mobile app uses Certificate Pinning, how would you still capture traffic in Charles?

Certificate Pinning means the app only trusts a specific server certificate, rejecting Charles' MITM certificate.
Ways to handle this:
- Disable/Bypass Pinning: Work with developers to add a debug flag to bypass pinning in test builds.
- Frida/Xposed/Objection: Use mobile instrumentation tools to bypass pinning.
- Root/Jailbreak Solutions: Patch the app to remove pinning libraries (last resort).

👉 In real projects, we usually request a test build without pinning to safely use Charles.

13) What steps will you take if you don't see HTTPS traffic in Charles after installing the SSL certificate?

- Check if SSL Proxying is enabled under Proxy > SSL Proxying Settings.
- Ensure the domain you're testing is added to the SSL Proxying list.
- Verify the device is using the correct proxy IP & port (8888).
- For Android 11+: enable 'Trust user-installed certificates' in Developer Options.
- For iOS: after installing the profile, go to Settings > General > About > Certificate Trust Settings → enable full trust.

👉 90% of the time, the issue is SSL Proxying not enabled for the domain.

14) How do you debug requests that are encrypted at the payload level (AES, RSA) inside HTTPS traffic?

Charles can decrypt HTTPS, but if the request body itself is encrypted:
- First, confirm if the app encrypts payload before sending.
- Compare with Postman calls where raw payload is available.
- Work with developers to get the encryption/decryption logic.
- Sometimes, we can attach the decryption library to Charles via External Tools or decode payload offline.

👉 Example: In a fintech app, OTP payloads were AES encrypted, and we used the encryption key (shared by devs) to decode them.

15) Can Charles capture WebSocket traffic? How do you analyze it?

Yes, Charles supports WebSocket (ws:// and wss://).
- Open Charles → look for the WebSocket session under Structure tab.
- Messages are shown as frames instead of HTTP requests.
- You can view incoming/outgoing frames, payload, and timestamps.
- Example: In a chat app, we validated that typing indicators and chat messages were reaching the backend in real time.

16) You see traffic from the browser but not from the mobile app. What do you check?

- Confirm the mobile device is on the same Wi-Fi as the Charles machine.
- Re-check mobile Wi-Fi settings → Proxy set to PC IP:8888.
- Ensure no VPN or other proxy is interfering.
- Reinstall SSL certificate on mobile if it's HTTPS traffic.
- Test with a simple API like http://example.com to see if proxying works.

17) What are the risks of using Charles Proxy in production?

- Sensitive data (passwords, tokens, credit card details) may be exposed.
- If proxy is left running, attackers could exploit it.
- MITM risk → device trusts Charles as CA, so malicious actors could abuse it.
- Logging personal data may violate GDPR/PCI compliance.

👉 Mitigation: Only use in test environments or with dummy accounts. Turn off proxy when not needed.

18) Breakpoints vs Rewrite vs Map Local vs Map Remote – when do you use each?

- Breakpoints: Manual, one-time edits (e.g., change payment amount before sending).
- Rewrite: Automates recurring modifications (e.g., replace USD → INR in every response).
- Map Local: Mock server response with a local file (e.g., return fake product catalog).
- Map Remote: Redirect calls to another server (e.g., staging → production).

👉 Rule of thumb:
- Quick one-time → Breakpoint
- Automated recurring → Rewrite
- Mock local response → Map Local
- Redirect server → Map Remote

19) How do you identify if an issue is frontend or backend using Charles logs?

- No request sent → frontend bug.
- Request sent, error from server (4xx, 5xx) → backend issue.
- Request successful but UI broken → frontend parsing/display issue.

Example: In an e-commerce app, inventory API returned correct data but product cards didn't render → frontend parsing issue.

20) How do you capture and validate Analytics events (Firebase, GA) in Charles?

- Enable SSL proxying for analytics domains (e.g., app-measurement.com, google-analytics.com).
- Trigger an event in the app (like Add to Cart).
- Verify request body → event name, parameters, timestamps.
- Check status = 200 OK, and confirm event data matches business rules.

👉 Example: We found that coupon codes weren't being logged in Firebase events, even though API worked fine.

21) How do you test multi-region APIs (e.g., India vs US) using Charles?

- Use Map Remote to redirect requests:
  api.india.myapp.com → api.us.myapp.com.
- Test app behavior when hitting different regional servers.
- Validate response headers (like Content-Language, Currency).

👉 In one project, we found that Indian users were still hitting US servers, causing latency.

22) How would you simulate fallback mechanisms when backend service is down?

- Use Rewrite Tool or Map Local to simulate server returning 500 Internal Server Error.
- Observe if the app falls back to cached data or shows proper error message.

Example: In a food delivery app, when the restaurant list API failed, app had to show 'Try again' instead of blank screen.

23) How would you debug an API call that never appears in Charles, even though the app is making it?

Possible causes and solutions:
- App might be using a direct socket connection (not HTTP/HTTPS) → Charles can't capture raw TCP.
- App may bypass system proxy (hardcoded IPs). Solution: use VPN-based tools or re-route via firewall.

- Certificate pinning → Charles certificate is rejected.
- DNS over HTTPS (DoH) → traffic bypasses Charles.

👉 First, confirm app uses HTTP/HTTPS and system proxy, then check pinning.

24) How does Charles handle HTTP/2 vs HTTP/3 traffic? Any limitations?

Charles fully supports HTTP/1.1 and HTTP/2 traffic.
- For HTTP/2, Charles can show multiplexed streams clearly.
- HTTP/3 (QUIC protocol over UDP) is not supported in most versions of Charles.
👉 If an app enforces HTTP/3, you may not see traffic in Charles. Solution: disable QUIC in the app or system.

25) Can Charles intercept DNS-over-HTTPS (DoH) requests?

No, Charles cannot intercept encrypted DNS-over-HTTPS directly.
👉 Workarounds:
- Disable DoH in system/browser.
- Use Wireshark or lower-level packet sniffers.
- Redirect DNS to normal port 53 for debugging.

26) How do you detect and handle apps that try to detect Charles Proxy usage (anti-proxy detection)?

- Some apps detect system proxy settings or untrusted certificates.
- To bypass:
  - Use VPN-based proxying instead of manual Wi-Fi proxy.
  - Patch app with Frida/Xposed to ignore proxy checks.
  - For SSL pinning, use test builds with pinning disabled.

27) If the app uses mutual TLS (mTLS), how does it affect Charles traffic inspection?

With mTLS, both client and server exchange certificates. Charles cannot impersonate the client cert.
👉 Options:
- Install client certificate in Charles.
- Work with devs to provide a debug cert for Charles.
- If not possible, Charles cannot decrypt traffic.

28) Can Charles Proxy be used to measure performance bottlenecks? How?

Yes. Charles shows request duration, DNS resolution, SSL handshake time, and response time.
- Look at 'Overview' tab for timing breakdown.
- Identify slow APIs by high response times.
- Helps isolate whether delay is DNS, SSL handshake, or server-side.

29) How would you simulate packet loss or intermittent connectivity using Charles?

- Use Throttling settings in Charles.
- Enable 'Throttle Settings' → Advanced → introduce latency, bandwidth limits, and packet loss %.

👉 Example: Simulate 20% packet loss to test retry logic in payment API.

30) An app uses GraphQL APIs instead of REST. How would you debug queries in Charles?

- GraphQL uses POST requests, usually to /graphql endpoint.
- The query/mutation is in the request body.
- In Charles, inspect JSON payload for 'query' and 'variables'.
- Modify payload with breakpoints to test different queries.

31) What steps would you take if the app is using WebRTC streaming – can Charles capture it?

WebRTC streams use UDP → Charles cannot intercept raw UDP.

👉 Charles can only capture the signaling API requests (like STUN/TURN negotiation), not the media stream.

For media, use Wireshark or browser dev tools.

32) How can you integrate Charles Proxy with Postman for replaying captured requests?

- Export request as cURL from Charles (right-click → Copy as cURL).
- Import cURL into Postman.
- Useful to replay the exact request with same headers and body.

33) How do you compare Charles logs with server logs to pinpoint an issue?

- Export Charles session as .chls or HAR.
- Compare request IDs, timestamps, and payloads with server logs.
- If server log shows request but app failed → frontend issue.
- If request never reached server → network/proxy issue.

34) Can Charles Proxy logs be converted into HAR or cURL files for automation?

Yes.

- Right-click request → Export/Copy as HAR or cURL.
- HAR can be fed into performance tools like Lighthouse.
- cURL can be used in scripts or Postman for regression.

35) You see CORS errors in the app but Charles shows 200 OK responses. How do you debug this?

CORS is enforced by the browser, not server.
- Charles shows raw server response (200 OK).
- Browser may block due to missing Access-Control-Allow-Origin headers.

👉 Use Charles to confirm headers are missing/misconfigured.

36) How do you handle APIs that use chunked transfer encoding in Charles?

- Charles supports decoding chunked responses.
- View response in 'Raw' or 'Text' tab.
- Helps verify if chunks arrive properly, and app assembles them correctly.

37) How would you detect and debug memory leaks or network floods using Charles?

- Watch for excessive repeated requests in Charles timeline.
- If requests loop indefinitely, it may indicate retry storm or memory leak in app.

👉 Example: Analytics events sent every 100ms due to bug, detected via Charles flood logs.

38) In a banking app, how would you validate two-factor authentication flows with Charles?

- Capture Generate OTP API (Step 1).
- Capture Validate OTP API (Step 2).
- Ensure session tokens issued only after OTP validation.
- Use Breakpoints to test wrong/expired OTPs.

39) In a ride-hailing app (like Uber), how would you debug real-time location updates using Charles?

- Location updates are usually sent as periodic POST/PUT requests.
- In Charles, filter by /location API.
- Validate payload → latitude, longitude, timestamp.
- Check frequency of updates matches expected interval.

40) In a video streaming app, how do you analyze DRM-protected requests (where payload is encrypted)?

- Charles can capture license acquisition API calls.
- You can't decrypt DRM payload, but:
  - Verify correct endpoints are hit.
  - Confirm status codes and license token exchange.
  - Validate error handling (403 Forbidden if license expired).

41)What is Charles Proxy and how have you used it in your project?

Real-Time Answer:

"Charles Proxy is a web debugging tool that allows us to monitor and analyze the network traffic (both HTTP and HTTPS) between client and server. I've used it extensively in my projects for debugging API calls, especially in mobile app testing.

In one of my projects, I was testing a fintech mobile app where we had to verify that the data sent from the app matched the expected backend request, and that the response was handled correctly in the UI. To do this, I connected my mobile device to Charles Proxy using Wi-Fi proxy settings and installed the Charles SSL certificate to view encrypted traffic.

I used it to:

- Intercept and analyse API calls in real time.
- Validate request parameters and headers, especially authentication tokens.
- Check if error responses like 401, 404, or 500 were due to backend or frontend issues.
- Use **breakpoints** to modify request payloads to test how the app responds to different data.
- Use **Map Local** feature to mock responses without waiting for backend fixes.
- Simulate network slowness or failure using **throttling**, to test app behavior in poor connectivity scenarios.

Charles Proxy helped us debug several issues early, especially around login failures, token expiration, and caching problems. It was especially useful for communicating API-related bugs clearly to the backend team, as I could export the Charles session logs and share them directly."

42) What is Charles Proxy and how have you used it in your e-commerce project?

Real-Time Answer:

"Charles Proxy is a web debugging proxy tool that captures HTTP and HTTPS traffic between a client (like a mobile app or browser) and a server. It allows testers to inspect, modify, and simulate API traffic. In my e-commerce project, I used Charles Proxy extensively during mobile app testing and API validation.

One key use case was during the checkout and payment flow, where I used Charles to:

intercept and inspect the Add to Cart, Apply Coupon, and Place Order API calls.

Ensure that sensitive data like card details were encrypted properly over HTTPS.

Confirm that the coupon code and discount logic matched the backend calculation.

Identify that a failed payment was due to a missing authorization header, which I quickly identified through Charles logs.

I also used Charles Proxy to:

Simulate different user journeys by modifying request payloads using breakpoints (e.g., testing orders with multiple shipping addresses).

Use Map Local to mock API responses for unavailable backend services, which helped us continue UI testing without waiting for fixes.

Detect caching issues—some APIs were returning stale inventory data; Charles helped prove that the Cache-Control headers were misconfigured.

Debug issues where product recommendations weren't loading properly; turned out the API was returning valid data, but the frontend wasn't parsing the JSON properly.

In addition, I configured Charles on real mobile devices to capture traffic during production smoke testing (with test accounts), especially for high-risk features like payment gateway integration and coupon redemption.

Overall, Charles Proxy helped me identify and isolate API-related bugs faster, collaborate more effectively with backend teams, and ensure a smooth and secure shopping experience for end users."

43) How do you Test Push Notifications APIs in Charles Proxy

1. Understand the Push Flow

Push notifications are sent via APNs (Apple Push Notification Service) for iOS or FCM (Firebase Cloud Messaging) for Android.

- The app doesn't directly talk to APNs/FCM. Instead:
  1. Your app's backend sends a request → APNs/FCM.
  2. APNs/FCM delivers the push → user's device.
- Charles can capture the API requests your app makes to its backend (e.g., to register a device token, trigger a notification, etc.).

2. Capture Device Registration

When the app installs/opens, it usually calls something like:

POST /registerDevice
Body: { "deviceToken": "xxxx", "platform": "android" }

---

3. Trigger a Push from Backend

Use your backend admin panel or API to send a test push notification.
Example request:

POST /sendPush
Body: { "deviceToken": "xxxx", "message": "Test Push" }

Charles will capture this request if you trigger it from your test device.

4. Using Breakpoints (Optional)

- Set a Breakpoint on the /sendPush API.
- Modify the message in Charles before it reaches the server.
  - Example: change "Test Push" → "Promo: 50% OFF".
- This lets you test different push content without changing the backend.

5. Validate Push Delivery

After the API succeeds, the push should appear on the device. Verify:
- Correct device received the push.
- Message/body is correct.
- Edge cases (long messages, special characters, etc.).

6. Simulate Errors

Using Charles Rewrite, simulate scenarios like:
- 401 Unauthorized → expired auth token.
- 400 Bad Request → invalid device token.

This helps test error handling in the app when push delivery fails.

7. Save Logs for Debugging

- Save the captured session (.chls file).
- Share it with developers if pushes aren't reaching the device.

☑ Summary

Charles Proxy cannot capture APNs/FCM direct traffic (since it's encrypted), but it's very useful for debugging your backend → push server requests (device registration, push triggers, error handling).

44) How to Test OTP APIs in Charles Proxy

1. Understand the OTP Flow

Typical OTP flow in an app:

1. User enters mobile number/email → App calls **Generate OTP API**.
2. Server sends OTP via **SMS / Email**.
3. User enters OTP → App calls **Validate OTP API**.
4. Server responds with success/failure.

Charles helps capture **step 1 & 3 API calls**.

2. Capture Generate OTP API

- Perform login/registration in your app.
- In Charles, look for an API call like:

POST /generateOTP

Body: { "phone": "9876543210" }

Response: { "status": "success", "otpId": "12345" }

☑ Verify:

- The phone/email sent in the request is correct.
- The server response includes an OTP ID or reference.

3. Capture Validate OTP API

- Enter the OTP received (SMS/Email).
- Charles will show:

POST /validateOTP

Body: { "otpId": "12345", "otp": "6789" }

Response: { "status": "success", "authToken": "abc123xyz" }

☑ Verify:

- OTP sent correctly in the request body.
- Response contains success and an authentication token/session.

4. Using Breakpoints

- Set a breakpoint on `/validateOTP`.
- Try entering a **wrong OTP** (e.g., modify 6789 → 1111).
- Observe how the server responds:
  - `400 Invalid OTP`
  - `401 Expired OTP`

👉 This helps test **negative scenarios** without waiting for real SMS/email.

5. Simulate Errors with Rewrite Tool

- Use Charles **Rewrite Tool** to mock server responses:
  - Change success → failure.
  - Delay response (simulate slow SMS gateway).
- Helps test app behavior when OTP API fails.

Example Rewrite Rule:

- Match: `/validateOTP`
- Response Body → `{ "status": "failure", "reason": "OTP expired" }`

6. Save Session for Debugging

- Export the `.chls` file with captured OTP API traffic.
- Share with developers when login/OTP issues occur.

☑ Key Takeaways

- Charles Proxy cannot capture the **actual SMS OTP** (since it comes via carrier), but it captures **backend OTP API calls**.
- You can test:
  - **Generate OTP request**
  - **Validate OTP request/response**
  - **Error handling** (wrong/expired OTP)
  - **Performance** (delay, retries)