Dynatrace Synthetics: Summary of Key Concepts and Configurations

Dynatrace Synthetics offers a proactive, highly detailed approach to monitoring application performance and availability through simulated user interactions. By executing synthetic tests from public or private locations, organizations can identify potential performance issues before they impact real users. This solution supports both frontend (browser-based) and backend (API-based) monitoring and is tightly integrated with Dynatrace's broader observability platform and its AI engine, Davis.

---

1. Overview and Architecture

At its core, Dynatrace Synthetic Monitoring simulates interactions across applications and APIs to ensure uptime, adherence to SLAs, and performance benchmarking. Its browser and HTTP monitors simulate user journeys and API behavior, respectively.

Key use cases include:

- **Uptime Validation** to confirm availability from various locations.
- **SLA Monitoring** to ensure service expectations are met.
- **Benchmarking** to measure performance against time, regions, or peers.

Key benefits:

- Early detection of problems before end users are affected.
- Broad coverage using global and internal (private) monitoring.
- In-depth diagnostics through waterfall charts and trace analysis.
- Integration with Davis AI for intelligent root cause analysis.

Architecture includes two monitor types:

- **Browser Monitors** use a real browser to simulate end-user interactions.
- **HTTP Monitors** focus on API health, chaining, and backend validations.

These monitors execute from:

- **Public Synthetic Locations**: Dynatrace-managed nodes across the globe.
- **Private Synthetic Locations (PSL)**: User-deployed agents for internal or restricted environments.

---

2. Monitor Configuration and Scheduling

Monitor configuration in Dynatrace is intuitive and powerful. It begins by defining what to monitor—either a frontend page or a backend API—and continues with setting up how often and where to run the tests.

Single-URL Browser Monitors simulate loading a single webpage and are ideal for tracking availability or homepage performance.

HTTP Monitors allow chaining multiple steps and testing complex workflows like authentication, data retrieval, and response validation. These support headers, payloads, tokens, and status code checks.

Choosing locations and frequency:

- Use **public locations** for customer-facing apps.
- Use **PSLs** for staging or internal services.
- Set **frequencies** based on the criticality of the monitored resource (every 1–15 minutes is common).

---

3. Thresholds, Metrics, and SLA Compliance

Once configured, synthetic monitors generate performance metrics such as:

- **Availability** (percent of successful executions),
- **Response Time** (across all steps),
- **Error Rate** (failures due to timeouts, bad status codes, etc.).

Thresholds can be manually defined or auto-detected using AI. For example, alerts can trigger if:

- Response time > 3s
- Availability < 99% over the past hour
- HTTP status ≠ 200

SLA Monitoring involves:

1. Defining expected thresholds.
2. Displaying results on custom dashboards.
3. Setting alerts on breaches.
4. Exporting SLA reports for stakeholders.

---

4. Clickpath Monitors and User Journeys

Clickpath Monitors simulate full user journeys involving multiple steps (e.g., login, search, add-to-cart). These are recorded using a browser-based recorder which logs each interaction. Each step can be renamed and validated.

Assertions confirm page states after each step using:

- String matches (e.g., “Welcome, John”),
- CSS selectors or XPaths,
- URL verifications.

Handling dynamic elements is done through wait actions, smart selectors, and token reuse, ensuring that monitors remain robust even when the UI changes or values are session-based.

---

5. API Monitoring and Advanced HTTP Configurations

HTTP Monitors are essential for backend and API availability validation. These support:

- REST and SOAP protocols,
- Response code validation,
- Content inspection (e.g., specific JSON values),
- Latency analysis by breakdown (DNS, SSL, TTFB).

Advanced configurations include:

- **Chaining HTTP requests** to simulate multi-step API workflows.
- **Token-based authentication**, passed securely via the credential vault or secret managers like AWS.
- **Managing dependencies** by extracting values from responses and using them in subsequent requests, simulating stateful API behavior.

---

6. Troubleshooting and Root Cause Analysis

Synthetic monitor failures may stem from application issues, network problems, or misconfigurations. Dynatrace provides a set of tools to isolate and resolve these.

Failure analysis involves:

- Reviewing logs and identifying failed steps.
- Checking if issues are location-specific or global.
- Analyzing timing metrics to see where latency occurs.

Davis AI helps automate RCA by correlating synthetic monitor failures with infrastructure events, service degradation, and RUM data. It highlights anomalies and impacted services, suggesting likely causes.

RUM Integration ensures that synthetic failures are viewed in context—whether real users were affected or not.

Diagnostic tools include:

- Execution logs,
- Waterfall charts (showing DNS, SSL, content load),
- Session traces.

---

7. Advanced Configurations and Reporting

Advanced capabilities enhance control and reporting.

Private Synthetic Locations (PSLs) are used for:

- Monitoring behind firewalls,
- Pre-production validation,
- Testing intranet portals,
- Data center performance comparisons.

To deploy a PSL:

- Install an ActiveGate,
- Enable the synthetic module,
- Register with Dynatrace,
- Assign monitors to the new location.

Alerting & Reporting:

- Alerts can be triggered via email, Slack, ServiceNow, PagerDuty, or Webhooks.
- Custom dashboards visualize availability, failures, and SLA breaches.
- Reports can be exported as PDFs or CSVs or fetched via Dynatrace APIs.
- Scheduled exports and dashboards keep stakeholders informed.

Dashboard metrics may include:

- builtin:synthetic.browser.availability,
- builtin:synthetic.http.response.time,
- API success rates, execution counts, and SLA widgets.

---

8. Capstone Simulation

A capstone exercise brings together all concepts through hands-on simulation. It includes:

- Building a **multi-step clickpath monitor** (login to checkout),
- Creating an **API monitor with chained requests** (token → data → validation),
- Configuring a **private synthetic location** using ActiveGate,
- Triggering failures intentionally and using **Davis AI for RCA**,
- Developing **dashboards and exporting reports** to summarize findings and performance trends.

This simulation mirrors enterprise scenarios and enables participants to demonstrate competence in full-stack synthetic monitoring with Dynatrace.

Conclusion

Dynatrace Synthetics empowers teams to ensure continuous availability, validate performance, and troubleshoot proactively. From simple URL tests to complex multi-step user journeys and chained API validations, Dynatrace offers a flexible, AI-enhanced toolkit for modern observability. With support for custom dashboards, deep diagnostics, and integrations into alerting workflows, synthetic monitoring becomes a foundational element of digital experience assurance.