Dynatrace Synthetics Overview & Architecture

---

1. Introduction to Dynatrace Synthetics

Dynatrace Synthetic Monitoring is a proactive monitoring capability that uses automated tests to simulate user behavior on applications and APIs from global or private locations. It ensures application availability and performance even when there are no real users interacting with the system.

- **Use Cases**
  - **Uptime Validation**: Confirm that your website or service is up and running.
  - **SLA Monitoring**: Ensure compliance with service-level agreements by tracking response time, availability, and error rate.
  - **Benchmarking**: Compare performance across time, regions, or against competitors.
- **Key Benefits and Value Proposition**
  - **Early issue detection** before users are affected.
  - **Global perspective**: Monitor from various geographic regions using public locations.
  - **Continuous testing** of APIs and user journeys without needing real traffic.
  - **Deep diagnostic insights** (via waterfall analysis, screenshots, and HTTP traces).
  - **Integration with Davis AI** for automatic root cause analysis.
  - **Supports both APIs (HTTP Monitors)** and **UI Journeys (Browser Monitors)**.

---

2. Architecture Overview

Dynatrace Synthetics is based on a distributed architecture that includes monitor types and execution environments.

- **Monitor Types**
  1. **Browser Monitors**
     - Simulate real-user interactions using Chromium-based browser engines.
     - Support single URL and clickpath-based testing.
     - Useful for frontend performance testing and journey validation.
  2. **HTTP Monitors**
     - Scripted or simple tests to validate RESTful APIs or web endpoints.
     - Support request chaining, authentication, and header customization.
     - Ideal for backend/API health and SLA checks.

Execution Locations

1. **Public Synthetic Locations**
    - Managed by Dynatrace.
    - Available in ~70+ global cloud locations.
    - Useful for monitoring from end-user geography and benchmarking performance.
2. **Private Synthetic Locations (PSL)**
    - Self-managed environments deployed via **ActiveGate**.
    - Ideal for internal, pre-production, or intranet application testing.
    - Enable synthetic testing in isolated or secured environments.

🔸 **Public vs Private – When to Use?**

Use Case	Use Public	Use Private
External customer app monitoring	✅	❌
Internal enterprise apps	❌	✅
SLA validation for global access	✅	❌
Testing apps behind firewall	❌	✅

3. Waterfall Analysis Basics

Waterfall analysis provides a visual breakdown of how individual page resources (HTML, CSS, JS, images, fonts, etc.) load over time.

🔹 **Resource Loading Sequence**

- Shows the exact order and duration for each resource's loading (DNS lookup, connection, SSL, request, response, rendering).
- Helps identify slow-loading elements.
- Understand dependencies between assets (e.g., how JS blocking affects rendering).

🔹 **Identifying Performance Bottlenecks**

- **DNS Lookup Time**: Long DNS times may indicate DNS issues.
- **Connection & SSL Handshake**: Delays can point to network or certificate issues.
- **Time to First Byte (TTFB)**: High values suggest server-side delays.
- **Blocking or Queuing**: Indicates render-blocking JavaScript or styles.

- **Large Payloads**: Image or video size issues.

Dynatrace enables drill-down from failed or slow synthetic steps directly into the waterfall to pinpoint root causes.

Synthetic Monitor Configuration

In this chapter, we focus on how to configure synthetic monitors in Dynatrace. This includes step-by-step guidance for creating different types of monitors, selecting suitable execution locations, scheduling test frequencies, and setting performance thresholds aligned with service level agreements (SLAs).

Creating Synthetic Monitors

Synthetic monitors in Dynatrace are configured to simulate user activity and API interactions. Depending on the monitoring goal—frontend experience or backend service uptime—you can choose between browser monitors and HTTP/API monitors.

1.1 Single-URL Browser Monitor

A Single-URL Browser Monitor is the simplest form of a browser-based test. It loads a single web page using a real browser engine and evaluates performance and availability from specified locations.

Steps to Create:

1. Navigate to **Synthetic Monitoring** in the Dynatrace UI.
2. Click on **Create a browser monitor**.
3. Enter the target **URL** (e.g., https://example.com).
4. Choose one or more **monitoring locations** (public or private).
5. Set **execution frequency** (e.g., every 5 minutes).
6. Configure optional alert thresholds.
7. Save and deploy the monitor.

Use Cases:

- Homepage performance monitoring
- Login page availability
- Page load behavior across geographies

1.2 API and HTTP Monitors

HTTP Monitors are used to test API endpoints or web services without browser interaction. These tests support chained requests, authentication, and response content validation.

Steps to Create:

1. Navigate to **Synthetic Monitoring** > **Create HTTP monitor**.
2. Define one or more **HTTP request steps** (GET, POST, etc.).
3. Specify **headers**, **payloads**, and **authentication** if needed.
4. Set **validation rules** (e.g., response code = 200, content must contain a string).
5. Choose **execution locations**.
6. Set **monitor frequency**.
7. Enable failure alerting if required.

Use Cases:

- REST API uptime testing
- Authentication token validation
- SLA monitoring of critical endpoints

2 Choosing Appropriate Locations and Scheduling Tests

Selecting execution locations and scheduling frequency are vital for effective monitoring.

2.1 Choosing Locations

- **Public Synthetic Locations**: Ideal for customer-facing applications; provide global performance insight.
- **Private Synthetic Locations**: Necessary for internal or pre-production environments.

Factors to consider:

- End-user geography
- Data compliance requirements
- Application exposure (public vs internal)

2.2 Scheduling Frequency

- Common intervals: 1 min, 5 min, 15 min
- High-frequency for critical services
- Lower frequency for non-critical checks or cost savings

3 Performance Metrics and Thresholds

Once monitors are running, you must define thresholds to detect anomalies and trigger alerts.

3.1 Key Metrics Tracked

- **Availability**: Percentage of successful executions
- **Response Time**: Measured in milliseconds or seconds
- **Error Rate**: Number of failed executions due to HTTP status or content mismatches

3.2 Threshold Configuration

Thresholds help determine when a monitor should be marked as degraded or failed.

Examples:

- Response time > 3 seconds = Warning
- Availability < 99% over 1 hour = Failure
- HTTP status code ≠ 200 = Error

You can define static thresholds or use Dynatrace's AI-based anomaly detection.

4 SLA Monitoring with Synthetics

SLAs define the expected performance and availability of services. Synthetic monitors provide consistent metrics to validate SLA compliance.

Steps to Monitor SLA:

1. Define SLA criteria (e.g., 99.9% availability, response time < 2s).
2. Use Dynatrace dashboards to display historical monitor results.
3. Set up custom alerts for breaches.
4. Export SLA reports via Dynatrace API or PDF dashboards.

Multi-Step Clickpath and User Journeys

In this chapter, we explore the configuration and usage of Clickpath Monitors in Dynatrace. These monitors simulate complex multi-step user journeys across web applications. Such scenarios are essential to ensure that mission-critical user interactions like logins, product searches, and checkouts function seamlessly from various global locations.

---

1 Clickpath Monitor OverviewA Clickpath Monitor simulates a sequence of user actions in a browser environment. These steps may include typing text, clicking buttons, navigating between

pages, and verifying content. Dynatrace uses a Chromium-based engine to accurately emulate browser behavior.

Key Features:

- Simulates real-world multi-step workflows
- Captures screenshots and timing at each step
- Detects UI and navigation issues proactively
- Supports both public and private synthetic locations

2 Simulating Complex Multi-Step User Journeys

Clickpath Monitors are commonly used to simulate:

- Login workflows (enter username, password, click login)
- Product or service searches
- Adding items to cart and checkout
- Form submissions and confirmation pages

Step-by-Step Flow Creation

1. Navigate to **Synthetic Monitoring** > **Create Browser Monitor**.
2. Choose **Clickpath browser monitor**.
3. Launch the **Clickpath Recorder** tool (built-in browser).
4. Perform the user journey manually (e.g., type in login details, click submit).
5. The recorder automatically logs each interaction as a step.
6. Optionally rename steps and insert wait times.
7. Set execution locations and frequency.
8. Save and deploy the monitor.

Clickpaths are ideal for identifying not just availability but functional correctness from an end-user perspective.

3 Assertions and Validations

Assertions are checks embedded in the monitor to validate that specific content appears or that the application behaves as expected at a given step.

1 Adding Content Checks

- After each step, add a **validation rule**:
    - Check for the presence of a string (e.g., "Welcome, John")

    - Match an element's CSS selector or XPath
    - Validate URLs or redirection paths

These checks help confirm that the page loaded properly and the workflow is progressing as designed.

3.2 Handling Dynamic Values and Redirects

Modern applications often involve dynamic content and automatic redirects. Dynatrace provides options to:

- Add **wait actions** to allow content to load
- Use **smart selectors** to handle changing element IDs
- Capture and **re-use values** across steps if required
- Accept or follow **redirects automatically** between steps

By incorporating assertions and adaptive actions, Clickpath Monitors become resilient to UI changes and provide accurate insight into user journey integrity.

API and HTTP Monitoring

This chapter focuses on HTTP Monitors within Dynatrace Synthetic Monitoring. These monitors are designed to validate the health and performance of web services such as REST APIs and SOAP endpoints. They provide lightweight, fast, and programmable ways to ensure backend service availability.

---

1 HTTP Monitor Overview

HTTP Monitors simulate API calls from various locations to check the responsiveness, correctness, and availability of backend services. Unlike browser monitors, they do not render UI elements, making them suitable for performance-sensitive backend validation.

Key Use Cases:

- Monitoring internal or external APIs
- Validating microservices communication
- Tracking service level indicators (SLIs) such as latency and availability

Protocols Supported:

- **REST APIs** (via GET, POST, PUT, DELETE, etc.)
- **SOAP services** (via HTTP XML-based requests)

2 Verifying Response Codes, Payloads, and Latency

Dynatrace HTTP Monitors provide fine-grained controls for verifying API response attributes:

Response Code Validation:

- Check that HTTP status codes are expected (e.g., 200 OK, 204 No Content).
- Define alerting conditions for 4xx and 5xx errors.

Payload Validation:

- Match the response body with expected content.
- Look for specific JSON keys or XML nodes.
- Validate content using regex patterns or string matches.

Latency Measurement:

- Measure total response time.
- Breakdown by DNS lookup, connection, TTFB, and response download time.
- Compare results with predefined performance thresholds.

3 Advanced Configurations

HTTP Monitors can be configured to simulate realistic, stateful API interactions and secure transactions.

3.1 Chaining HTTP Requests

- Combine multiple HTTP steps in a single monitor.
- Pass output from one step (e.g., session token) as input to another.
- Useful for multi-step workflows like authentication + data retrieval.

3.2 Adding Authentication Tokens

- Support for **Basic Authentication**, **Bearer Tokens**, **API Keys**, and custom headers.
- Securely inject credentials stored in Dynatrace's **Credential Vault** or fetched from external tools like AWS Secrets Manager.

3.3 Managing Dependencies Between Requests

- Use **extracted variables** from one response as input for the next step.
- Ensure correct order and conditional flows based on previous response status.
- Handle cookies, tokens, and dynamic session data.

Troubleshooting & Root Cause Analysis

This chapter focuses on identifying and resolving synthetic monitor failures using the robust diagnostic tools provided by Dynatrace. It also explores how Dynatrace AI (Davis) correlates synthetic issues with backend and real-user monitoring data to offer actionable root cause insights.

1 Identifying Synthetic Failures

Synthetic monitor failures can occur for various reasons, such as application unavailability, incorrect element selectors, network timeouts, or backend errors. Early detection and diagnosis are critical to minimizing downtime and impact.

1.1 Diagnosing Failures and Timeouts

- Review monitor execution logs to identify which step or request failed.
- Evaluate if the error is consistent across locations or intermittent.
- Inspect timeout values and response time to determine latency-induced failures.

1.2 Analyzing Error Codes and Slow Responses

- Identify HTTP error codes (e.g., 404 Not Found, 500 Internal Server Error).
- Monitor response codes for each step or API call.
- Analyze slow response times by breaking them into DNS resolution, SSL handshake, TTFB, and payload download.

2 Root Cause Analysis (RCA)

Dynatrace offers built-in AI capabilities (Davis AI) to help perform root cause analysis. When a synthetic monitor fails, Davis correlates synthetic data with infrastructure, application metrics, and real-user behavior.

2.1 Using Dynatrace AI Insights

- Davis AI highlights potential causes such as backend service degradation, high CPU usage, or database latency.
- AI insights appear as contextual cards directly within synthetic monitor failure details.
- RCA suggestions include impacted services, metrics anomalies, and related log entries.

2.2 Linking Synthetic Failures with Real User Monitoring (RUM)

- View affected user sessions during the same time period to determine real-user impact.
- RUM integration allows validation of whether synthetic failure mirrors actual user issues or isolated synthetic-only issues.
- Analyze conversion rate drops and bounce rates in correlation with synthetic degradation.

3 Troubleshooting Tools

Dynatrace provides several tools to assist in diagnostics and troubleshooting:

3.1 Viewing Logs and Traces

- Access synthetic monitor logs to view execution steps and error messages.
- Download HAR files or session logs for external analysis.
- View traces related to HTTP requests when backend tracing is enabled.

3.2 Waterfall Analysis for Deep-Dive Diagnostics

- Waterfall charts provide visual breakdowns of browser-based monitor steps.
- Analyze load sequence: DNS lookup, TCP connection, SSL handshake, and request/response durations.
- Identify slow-loading resources, render-blocking scripts, and third-party delays.
- Compare fast vs slow runs for the same clickpath to isolate variables.

Advanced Configurations & Reporting

This chapter explores advanced synthetic monitoring configurations and how to leverage Dynatrace's alerting, reporting, and visualization capabilities for actionable insights and performance governance.

---

1 Private Synthetic Locations

Private Synthetic Locations (PSLs) allow organizations to monitor internal applications behind firewalls or hosted within restricted environments. They are deployed using the Dynatrace ActiveGate component.

1.1 Deploying and Configuring Private Monitoring Locations

- Install a **Private ActiveGate** on a VM or physical machine within the internal network.
- Ensure outbound connectivity to the Dynatrace cluster for test result transmission.
- Assign the ActiveGate to host **synthetic-enabled modules**.
- Validate deployment by creating a test monitor and selecting the PSL as an execution location.

1.2 Use Cases for Internal Applications

- Monitoring **intranet portals** and **internal APIs**
- Pre-production and **staging environment validation**
- SLA assurance for services **not exposed to the public internet**
- Monitoring across multiple data centers or **hybrid-cloud topologies**

2 Alerting and Reporting

Dynatrace provides built-in mechanisms to trigger alerts and generate reports based on synthetic monitor results.

2.1 Setting Up Alerts and Notifications

- Navigate to **Settings > Anomaly detection > Availability**.
- Define alert conditions (e.g., monitor fails more than 3 times in 5 minutes).
- Configure alert destinations such as:
    - **Email**
    - **Slack**
    - **ServiceNow**
    - **PagerDuty**
    - **Webhooks**
- Utilize **custom event rules** to filter by monitor name, location, or tag.

2.2 Generating and Exporting Custom Reports

- Use **custom dashboards** and **Data Explorer** to create visual reports.
- Export data in formats such as CSV, PDF, or via **Dynatrace API v2**.
- Schedule report generation and distribution to stakeholders.
- Leverage SLA widget for real-time visibility into compliance targets.

3 Visualization and Dashboards

Dashboards are essential for communicating application health and synthetic performance trends.

3.1 Building Custom Dashboards

- Go to **Dashboards > Create Dashboard**.
- Add tiles for:
    - Uptime trends
    - Failed monitor count
    - Response time by location
    - API performance
- Use filters to segment data by tag, management zone, or application.

3.2 Adding Synthetic Metrics

- Include metrics like:
    - builtin:synthetic.browser.availability
    - builtin:synthetic.http.response.time
    - builtin:synthetic.browser.events
- Configure thresholds for coloring (green/yellow/red) based on SLA.
- Combine with real user metrics for full-stack visualization.

Capstone Exercise

This capstone project allows participants to synthesize their learning into a comprehensive, real-world simulation. Participants will configure full-stack synthetic monitoring, analyze failures, and generate reports—mirroring enterprise-grade monitoring practices.

---

1 Designing a Complete Synthetic Monitoring Setup

Multi-step Clickpath Monitor

- Design a monitor that simulates a user journey across a demo application.
- Include steps like login, product search, item selection, cart addition, and logout.
- Add validations at each step to confirm successful navigation (e.g., "Welcome user" or "Product listed").
- Configure execution from 3 public and 1 private location, running every 5 minutes.

API Monitor with Chained Requests

- Create an HTTP monitor with 3 chained steps:
    1. Authenticate and receive a session token.
    2. Use the token to request a list of resources.
    3. Validate the payload response for accuracy.
- Apply performance thresholds and content checks.

2 Configuring Private Synthetic Locations

- Deploy a Private ActiveGate on a local VM (Windows or Linux).
- Register it with the Dynatrace tenant and enable the synthetic module.
- Configure location name, tags, and availability zones.

- Assign monitors to run from this PSL and verify test results.

3 Root Cause Analysis and Troubleshooting

Simulating Failures

- Intentionally break a monitor (e.g., wrong element selector or incorrect endpoint).
- Allow it to fail from all locations.

Performing RCA Using Dynatrace AI

- Use Davis AI insights to determine:
    - What step or condition failed
    - Any backend dependencies affected
    - Related events or infrastructure issues
- Correlate synthetic issues with real-user sessions (if applicable)

4 Generating Reports and Configuring Alerts

Creating Dashboards with Synthetic Metrics

- Build a dashboard titled "Capstone Monitoring Overview"
- Include:
    - Synthetic availability per monitor
    - Failed executions (by type and location)
    - API response time trends
    - SLA widget

Exporting Performance Reports

- Create a custom report for leadership with:
    - Performance over 7 days
    - Failure summaries
    - Recommendations for improvement
- Export as PDF and schedule weekly email delivery