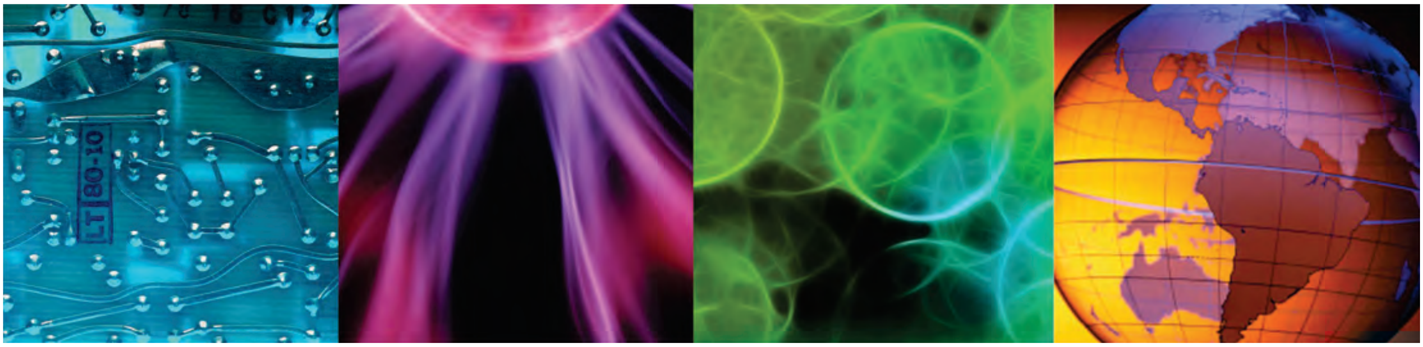
IBM Training

# IBM Application Performance Management Advanced 8.1.3 Fundamentals

## Course Guide

Course code TOD45 ERC 1.0

August 2016

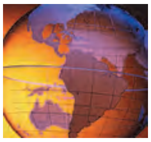


ibm.com/training

Authorized IBM | Training

The information contained in this publication is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this publication, it is provided AS IS without warranty of any kind, express or implied. In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this publication or any other materials. Nothing contained in this publication is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this publication to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth, savings or other results.



# Contents

# About this course

## IBM Application Performance Management 8.1.3 and Application Diagnostics Fundamentals

This course covers the key features of IBM Application Performance Management Advanced, including IBM Application Diagnostics, including configuration, resource monitoring, and transaction diagnosis. Exercises will be carried out for a WebSphere Application Server on a Linux system.

Node.js features are covered in slides, but also in hands-on exercises on a Linux system in Appendix A of the Course Exercises.The .NET and Ruby agents are covered in the slides. For information about other related courses, visit the Cloud & Smarter Infrastructure education training paths website:

ibm.com/software/software/tivoli/education/

| | Details |
|---|---|
| **Delivery method** | Self-paced virtual classroom (SPVC) |
| **Course level** | ERC 1.0 |
| | This course is a new course. |

| | Details |
|---|---|
| **Product and version** | IBM Application Diagnostics 8.1.3 |
| **Duration** | 1 day |
| **Skill level** | Intermediate |

# About the student

This course is designed for application-monitoring specialists and technical sales personnel.

Before taking this course, make sure that you have the following skills:

- Familiarity with application server monitoring
- Familiarity with the Performance Management Console

Before taking this course make sure that you have taken the following courses

- IBM Monitoring 8.1.3 Implementation and Administration (TM673)
- InterConnect 2015 CAP-3630: Introduction to Application Performance Management (recommended but not required)

# Learning objectives

## Objectives

In this course, you learn to perform the following tasks:

- Describe the IBM Application Performance Management Advanced architecture
- Learn the monitoring features of the four supported IBM Application Diagnostics agents
- Monitor resources, application code, transactions, and users with the WebSphere agent

# Course agenda

The course contains the following units:

1. Monitoring with IBM Application Performance Management Advanced

   The first presentation is technical overview of IBM Performance Management and Application Diagnostics.

   IBM Application Performance Management Advanced includes monitoring agents for several application platforms. As of this writing, four of those monitoring agents support code-level monitoring:

2. Monitoring application resources

   This presentation is a technical overview of the resource-monitoring features of IBM Application Diagnostics agents.

   In this lab session, you use IBM Application Performance Management to monitor WebSphere resources.

3. Code-level monitoring

   This presentation is a technical overview of the code-level monitoring features of IBM Application Diagnostics.

   Code-level diagnostic monitoring is available as of this writing for the agents that monitor these products:

   - WebSphere Application Server
   - Microsoft .NET
   - Node.js
   - Ruby

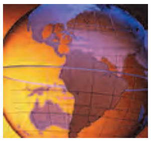
4. Transaction tracking

   This presentation is an overview of the transaction monitoring features of IBM Application Performance Management Advanced.

   In addition to resource monitoring and code-level diagnosis, IBM Application Performance Management Advanced Diagnostics supports transaction tracking.

5. Synthetic transaction and user monitoring

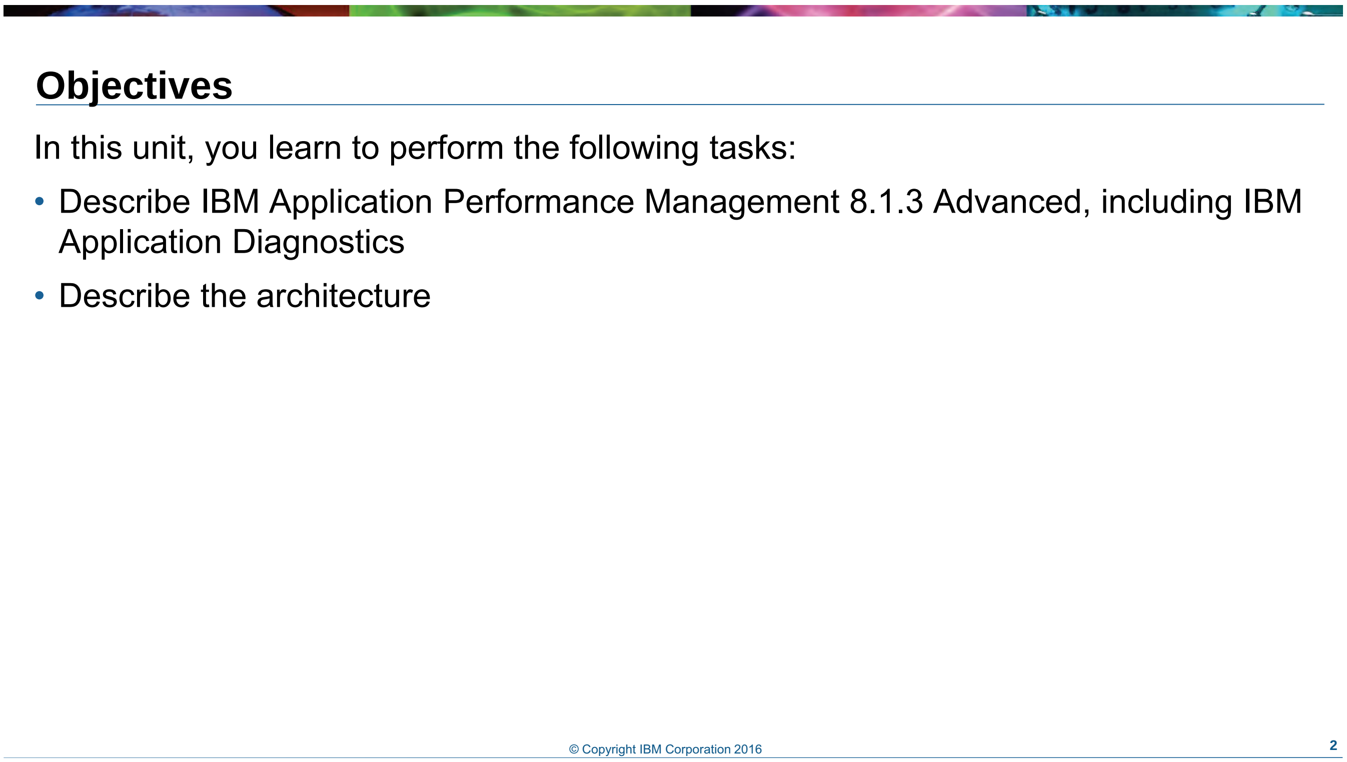
   This presentation is an overview of the transaction monitoring features of IBM Application Performance Management Advanced for synthetic transactions, websites, and users.



# 1 Monitoring with IBM Application Performance Management Advanced

1 Monitoring with IBM Application Performance Management Advanced

The first presentation is technical overview of IBM Performance Management and Application Diagnostics.



## Objectives

In this unit, you learn to perform the following tasks:

- Describe IBM Application Performance Management 8.1.3 Advanced, including IBM Application Diagnostics
- Describe the architecture

2

*Objectives*

In this unit, you learn to perform the following tasks:

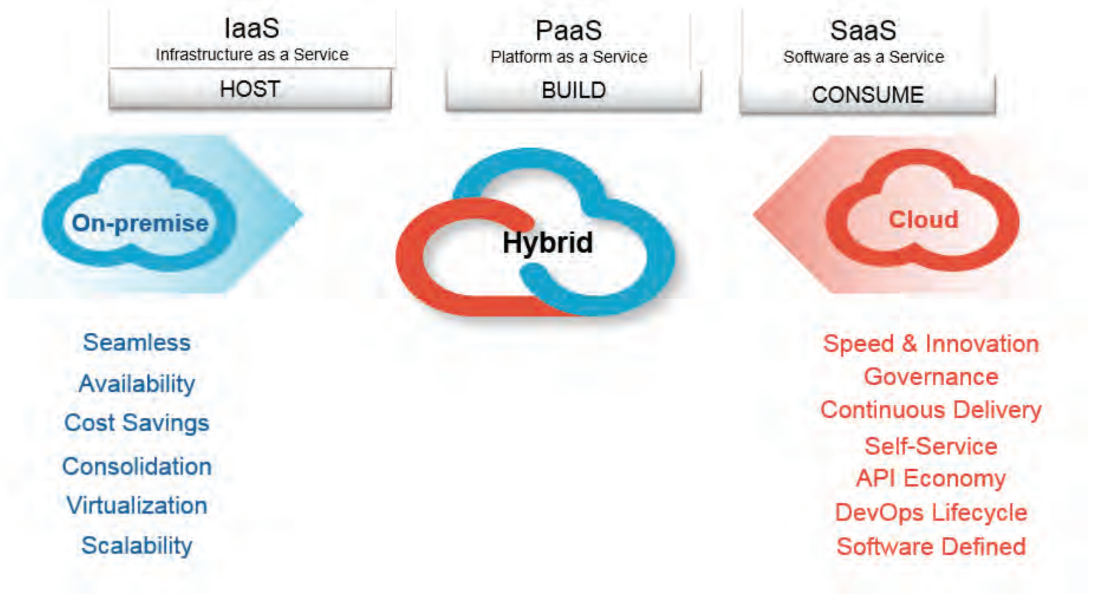
- Describe IBM Application Performance Management 8.1.3 Advanced, including IBM Application Diagnostics
- Describe the architecture

# Lesson 1  Overview and architecture

Lesson 1 Overview and architecture

In this section of this presentation, you get an overview of the IBM Performance Management Advanced solution, including the value proposition, portfolio, and resource coverage.

## The Cloud landscape



IaaS
Infrastructure as a Service
HOST

PaaS
Platform as a Service
BUILD

SaaS
Software as a Service
CONSUME

On-premise

Hybrid

Cloud

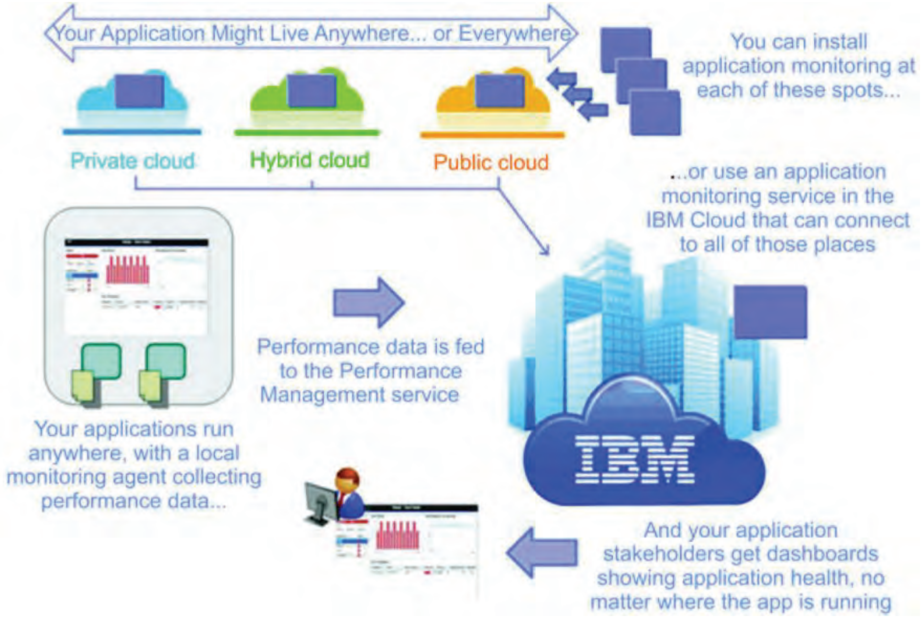
Seamless
Availability
Cost Savings
Consolidation
Virtualization
Scalability

Speed & Innovation
Governance
Continuous Delivery
Self-Service
API Economy
DevOps Lifecycle
Software Defined

*The Cloud landscape*

The cloud landscape alters where and how applications are built, deployed, and consumed. Platforms and software are moving from the traditional on-premises deployment to cloud based services for various reasons. With this change comes the need to monitor and manage the existing and new deployment models together with one solution.

## IBM Performance Management



You can install application monitoring at each of these spots...

...or use an application monitoring service in the IBM Cloud that can connect to all of those places

Performance data is fed to the Performance Management service

Your applications run anywhere, with a local monitoring agent collecting performance data...

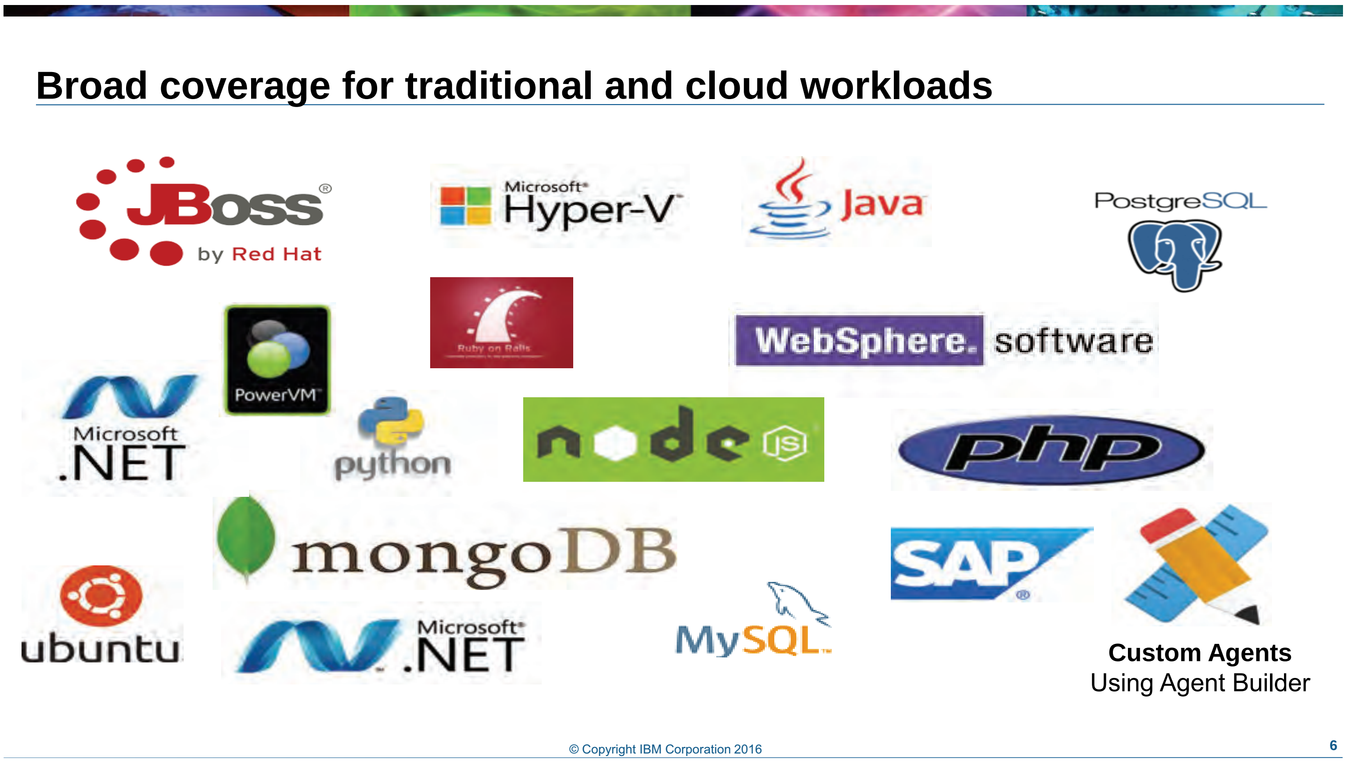
And your application stakeholders get dashboards showing application health, no matter where the app is running

5

*IBM Performance Management*

IBM Performance Management is a comprehensive solution that helps manage the performance and availability for complex applications that might be running in a data center, public cloud, or hybrid combination. This solution provides you with visibility of your applications, ensuring optimal performance and efficient use of resources.

The hands-on portion of this course uses an on-premises installation of IBM Application Performance Management Advanced. Cloud and hybrid environments are summarized in the presentations.



## Broad coverage for traditional and cloud workloads

JBoss by Red Hat
Microsoft Hyper-V
Java
PostgreSQL
PowerVM
Ruby on Rails
WebSphere software
Microsoft .NET
python
node js
php
mongoDB
SAP
ubuntu
Microsoft .NET
MySQL

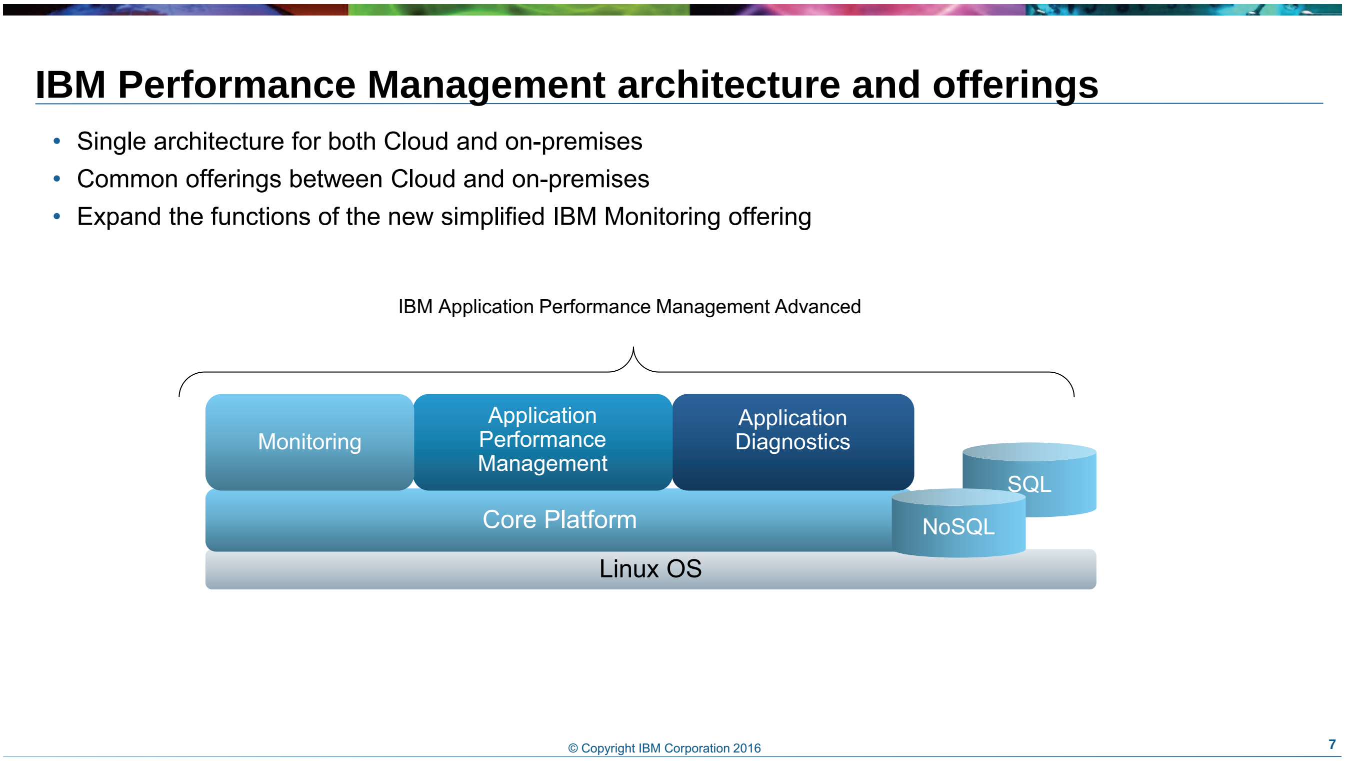
**Custom Agents**
Using Agent Builder

© Copyright IBM Corporation 2016

6

*Broad coverage for traditional and cloud workloads*

This slide illustrates the breadth of coverage offered by the IBM Performance Management portfolio for traditional and cloud workloads. Additional extension packs can be purchased to enhanced monitoring the Citrix Virtual Desktop Infrastructure, Hadoop, and the SAP HANA Database. Also available are IBM Website Monitoring on Cloud and Operations Analytics Predictive Insights.



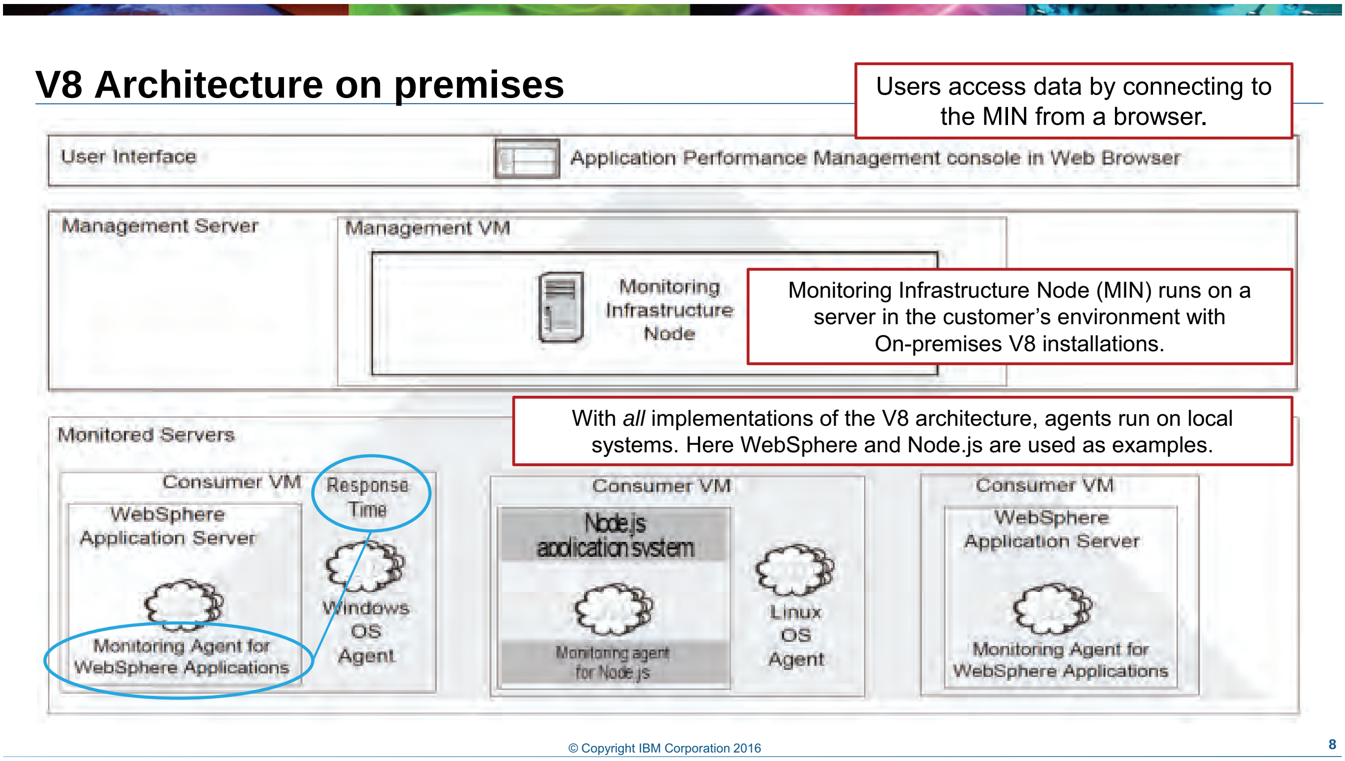
## IBM Performance Management architecture and offerings

- Single architecture for both Cloud and on-premises
- Common offerings between Cloud and on-premises
- Expand the functions of the new simplified IBM Monitoring offering

IBM Application Performance Management Advanced
Monitoring
Application Performance Management
Application Diagnostics
SQL
Core Platform
NoSQL
Linux OS

*IBM Performance Management architecture and offerings*

With IBM Performance Management, on-premises and cloud use a single architecture and common offerings.



## V8 Architecture on premises

Users access data by connecting to the MIN from a browser.

Monitoring Infrastructure Node (MIN) runs on a server in the customer's environment with On-premises V8 installations.

With *all* implementations of the V8 architecture, agents run on local systems. Here WebSphere and Node.js are used as examples.

User Interface
Application Performance Management console in Web Browser
Management Server
Management VM
Monitoring Infrastructure Node
Monitored Servers
Consumer VM
WebSphere Application Server
Monitoring Agent for WebSphere Applications
Response Time
Windows OS Agent
Consumer VM
Node.js application system
Monitoring agent for Node.js
Linux OS Agent
Consumer VM
WebSphere Application Server
Monitoring Agent for WebSphere Applications

8

*V8 Architecture on premises*

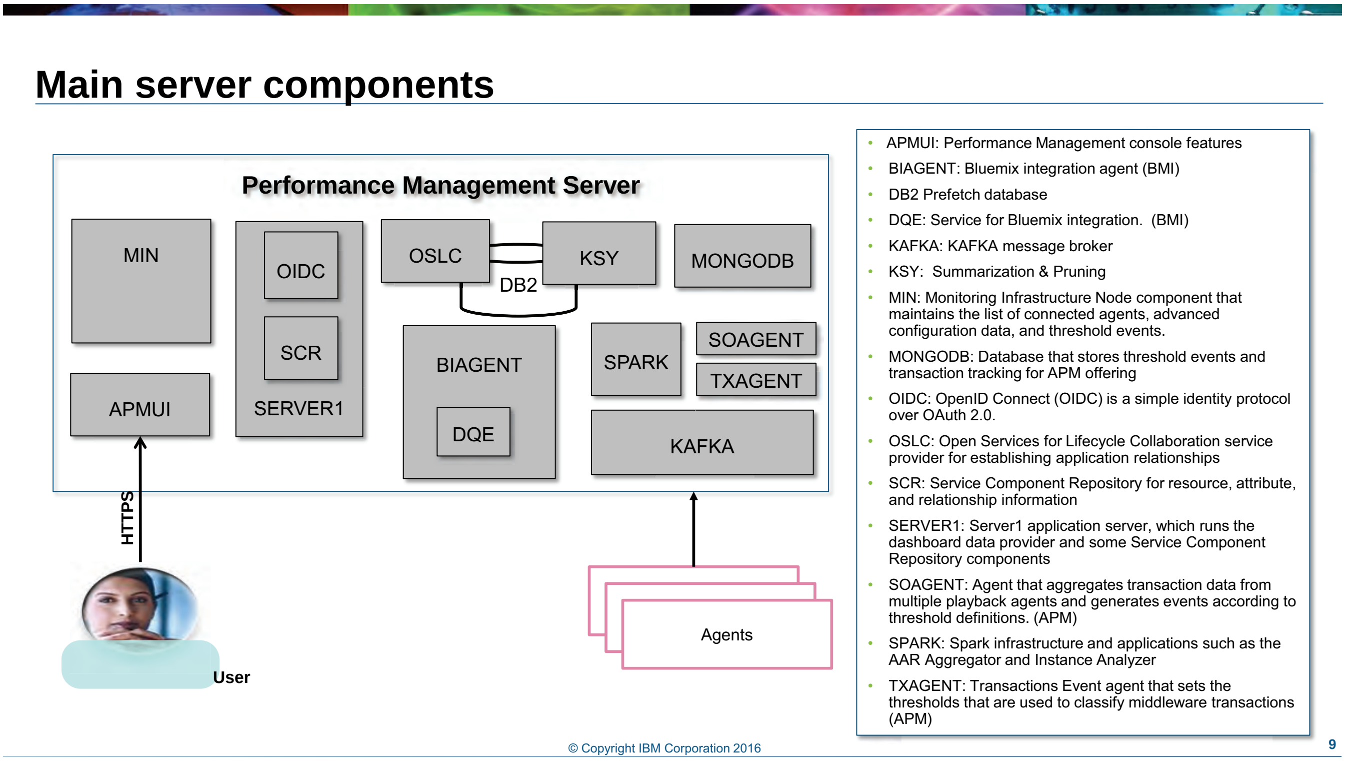
This slide illustrates the IBM Application Performance Management Advanced architecture, including IBM Application diagnostics.

At a high level, the Monitoring Infrastructure Node, or MIN, is the server component of the infrastructure. Agent installation packages are configured to be able to communicate with the MIN. You view data in the Performance Management console, at the top of the page.

The two left-most Consumer VMs illustrate what you need to make available all Application Diagnostics features for their respective application environments.

When properly configured, the Response Time agent interacts with the monitoring agent for WebSphere to support transaction topologies. As of the creation date of this presentation, the Node.js agent does not support topologies. Therefore, the Response Time agent is not installed on the Node.js system.

Transaction Topologies are not currently available on the system at the lower right, because the Response Time agent is not installed there.



## Main server components

Performance Management Server

MIN

OIDC

SCR

SERVER1

OSLC

DB2

KSY

MONGODB

APMUI

BIAGENT

DQE

SPARK

SOAGENT

TXAGENT

KAFKA

HTTPS

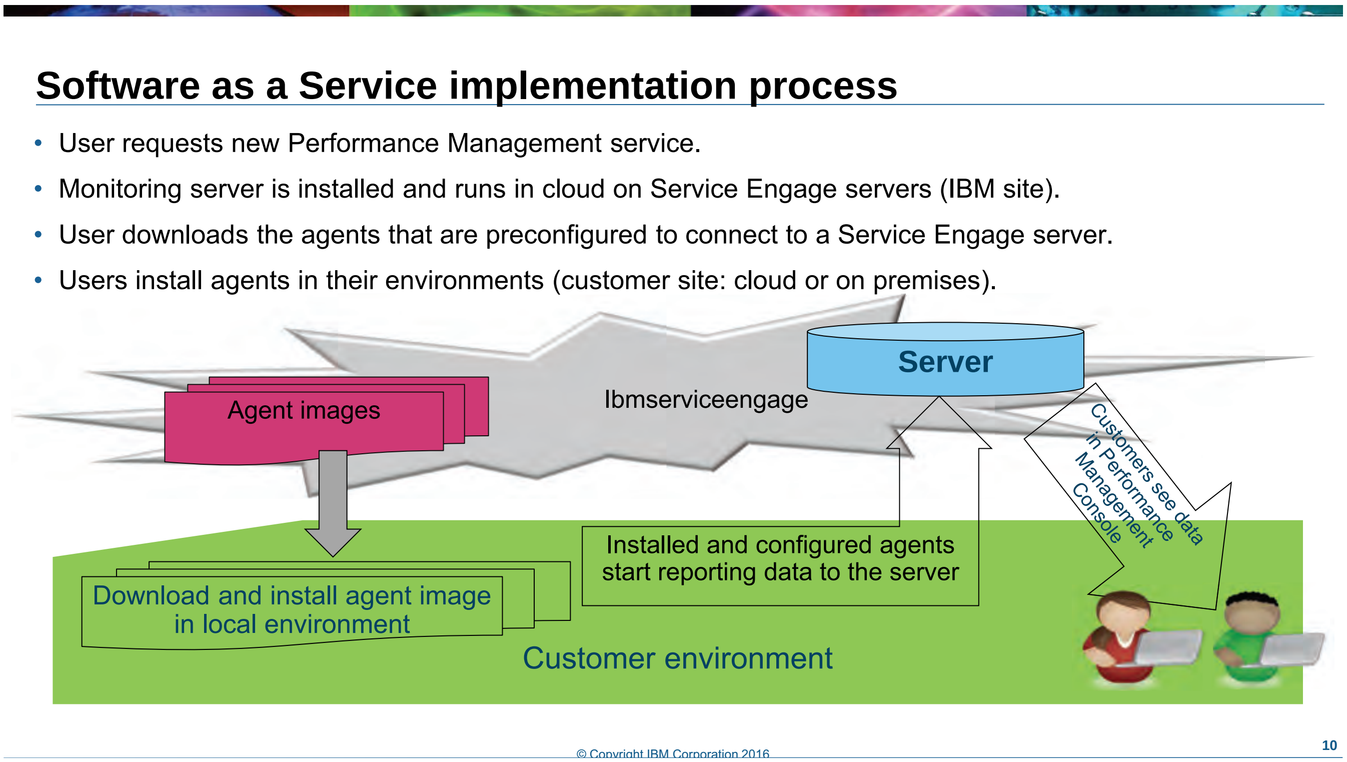
User

Agents

- APMUI: Performance Management console features
- BIAGENT: Bluemix integration agent (BMI)
- DB2 Prefetch database
- DQE: Service for Bluemix integration.  (BMI)
- KAFKA: KAFKA message broker
- KSY:  Summarization & Pruning
- MIN: Monitoring Infrastructure Node component that maintains the list of connected agents, advanced configuration data, and threshold events.
- MONGODB: Database that stores threshold events and transaction tracking for APM offering
- OIDC: OpenID Connect (OIDC) is a simple identity protocol over OAuth 2.0.
- OSLC: Open Services for Lifecycle Collaboration service provider for establishing application relationships
- SCR: Service Component Repository for resource, attribute, and relationship information
- SERVER1: Server1 application server, which runs the dashboard data provider and some Service Component Repository components
- SOAGENT: Agent that aggregates transaction data from multiple playback agents and generates events according to threshold definitions. (APM)
- SPARK: Spark infrastructure and applications such as the AAR Aggregator and Instance Analyzer
- TXAGENT: Transactions Event agent that sets the thresholds that are used to classify middleware transactions (APM)

*Main server components*

This slide lists and describes each of the main services of the Performance Management server. Users access the Performance Management console through the APMUI service.



## Software as a Service implementation process

- User requests new Performance Management service.
- Monitoring server is installed and runs in cloud on Service Engage servers (IBM site).
- User downloads the agents that are preconfigured to connect to a Service Engage server.
- Users install agents in their environments (customer site: cloud or on premises).

Server

Ibmserviceengage

Agent images

Customers see data in Performance Management Console

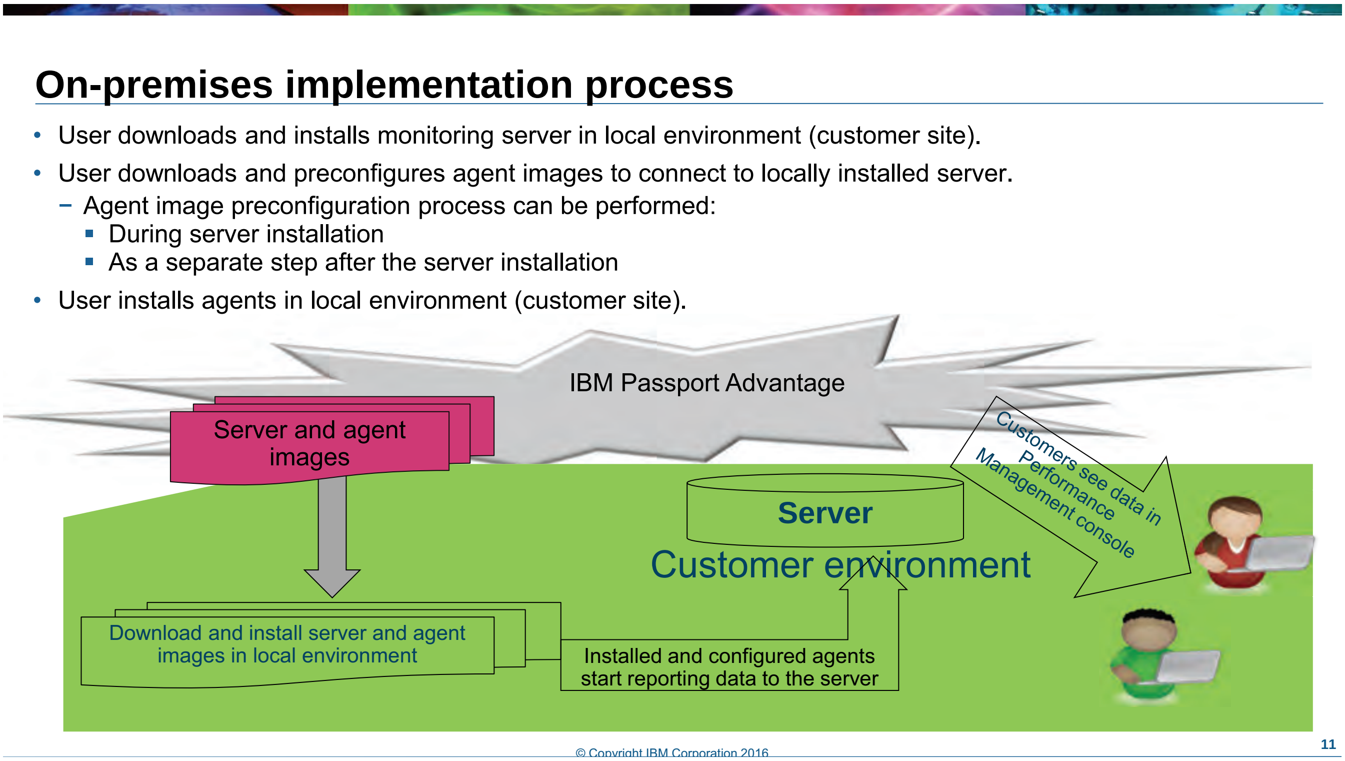
Installed and configured agents start reporting data to the server

Download and install agent image in local environment

Customer environment

*Software as a Service implementation process*

This slide shows the deployment process for IBM Performance Management as a service. The customer requests a new Performance Management service from IBM Service Engage. The Performance Management is deployed and runs in the cloud on the IBM Service Engage servers. The customer downloads the agents, which are preconfigured to connect to a server running on Service Engage. The customer installs the agents in the customer environment, be that on a server in the cloud or on-premises.



## On-premises implementation process

- User downloads and installs monitoring server in local environment (customer site).
- User downloads and preconfigures agent images to connect to locally installed server.
  - Agent image preconfiguration process can be performed:
    - During server installation
    - As a separate step after the server installation
- User installs agents in local environment (customer site).

IBM Passport Advantage

Server and agent images

Download and install server and agent images in local environment

Server

Customer environment

Installed and configured agents start reporting data to the server

Customers see data in Performance Management console

© Copyright IBM Corporation 2016

11

*On-premises implementation process*

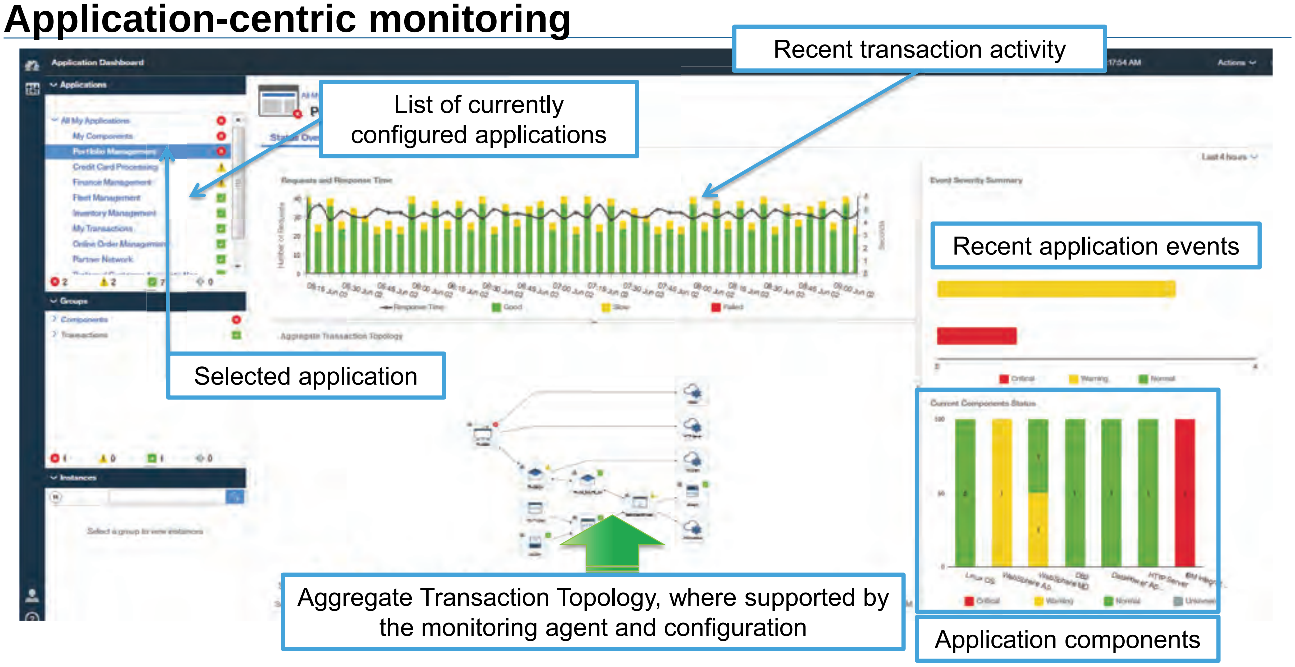
This slide shows the deployment process for IBM Monitoring on-premises. The customer purchases IBM Monitoring and can then download the software from IBM Passport Advantage®. Customers download and install Performance Management server in their local environment. Customers also download and preconfigure agent images to connect to a locally installed server. This preconfiguration process can be performed either during the server installation or as a separate step after the server installation. Customers can then install agents in their local environment.

# Lesson 2  Performance Management console

## Lesson 2 Performance Management console

In the second section of this presentation, you get a high-level overview of the IBM Performance Management console, particularly the key features of IBM Application Diagnostics.

## Application-centric monitoring



Recent transaction activity

List of currently configured applications

Recent application events

Selected application

Aggregate Transaction Topology, where supported by the monitoring agent and configuration

Application components

*Application-centric monitoring*

One of the key features of the version 8 architecture is the application-centric UI, which can take the operator from a symptom to a cause in just a few mouse clicks.

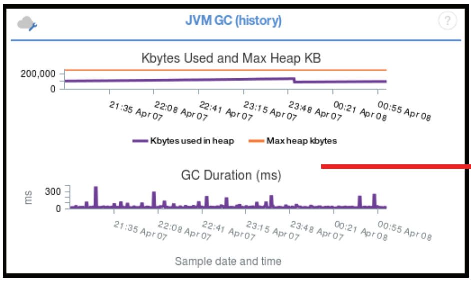
Clicking the Requests and Response Time widget exposes an End User Requests dashboard. Drilling down from the topology exposes instance topologies (the .NET agent in this example) and other response time features.

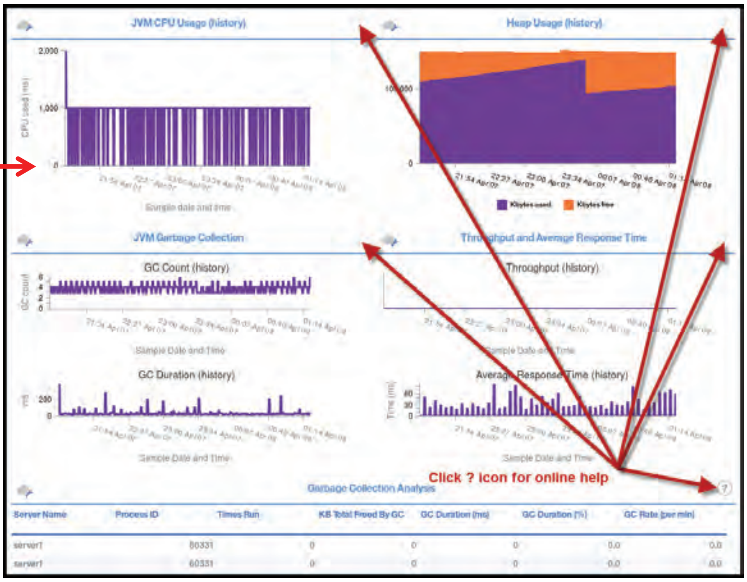
The dark red arrow inside the circle shows that the Portfolio Management application is selected from the list of configured applications.

Because application topologies are configured for this application, an aggregate application topology is shown in the middle of the dashboard (see the blue-green arrow).

A bar for each component type is displayed at the far right. Clicking one of those bars takes you to an instance dashboard for the relevant component type, such as Microsoft Exchange Server, as shown on the next slide.

## Resource dashboards: Here for garbage collection





Click ? icon for online help

| Server Name | Process ID | Times Run | KB Total Freed By GC | GC Duration (ms) | GC Duration (%) | GC Rate (per min) |
|---|---|---|---|---|---|---|
| server1 | 60331 | 0 | 0 | 0 | 0.0 | 0.0 |
| server1 | 60331 | 0 | 0 | 0 | 0.0 | 0.0 |

More detail dashboards available:

- WebSphere queries
- EJB containers
- Messaging and messaging engines
- Thread pools
- Database connection pools
- Web services

*Resource dashboards: Here for garbage collection*

Application Diagnostics also provides resource data, called Performance Monitoring Infrastructure (or PMI) data, in the case of WebSphere.

This slide illustrates the data that is available for JVM garbage collection, which is abbreviated GC on certain widgets. Some of the charts on this pane cover the past 24 hours, and some show the past 2 hours. Look carefully at the pane heading to understand the duration of the data being plotted.

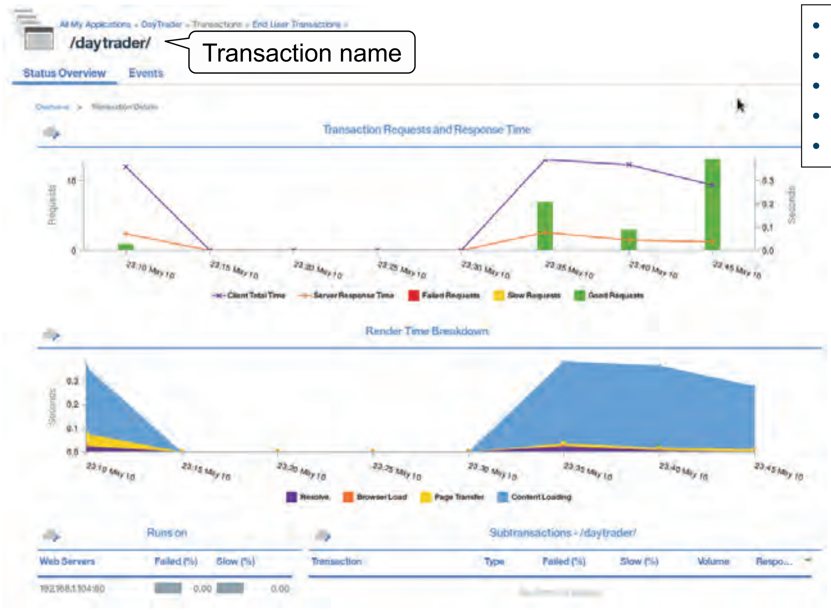
Three factors stand out on this view:

1. First is the increase in Java virtual machine processor usage on the right end of the JVM CPU Usage chart.
2. This increase corresponds to increased heap usage around the same time.
3. You also see a spike in the average response time. That might be worthwhile to investigate.

You can put the mouse pointer over the plot chart points and hold the left mouse key down. Move the mouse to the right or left, and a small window shows the value and the date and time it is recorded.

Garbage Collection is just one of the detail panes that you can invoke from the instance Overview pane. You can return directly to the Overview pane by clicking that hyperlink. When you hold your mouse pointer over a pane and see the hand icon, that is a link to another page.

## Viewing individual transaction details



| Web Servers | Failed (%) | Slow (%) |
|---|---|---|
| 192.168.1.104:80 | 0.00 | 0.00 |

| Transaction | Type | Failed (%) | Slow (%) | Volume | Respo... |
|---|---|---|---|---|---|

- Requests by status
- Average response time
- Render Time Breakdown
- System where transaction runs
- Subtransactions (not always present)

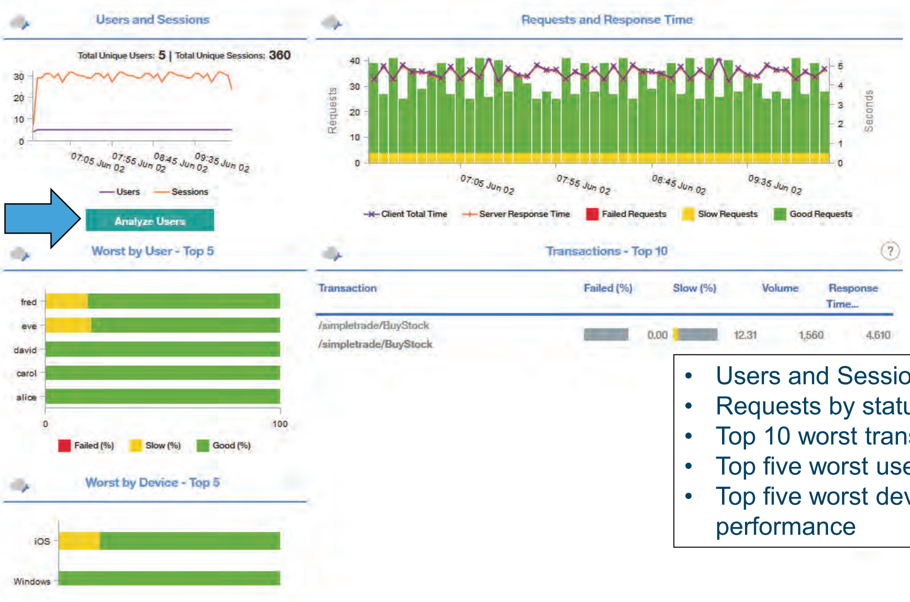
*Viewing individual transaction details*

The transaction details window shows requests by status and average response time for an individual transaction. It also shows you the server or servers that the transactions run on.

Also included is a render-time breakdown of requests, including these examples:

- Resolve time
- Browser load
- Page transfer
- Content loading

## User transactions



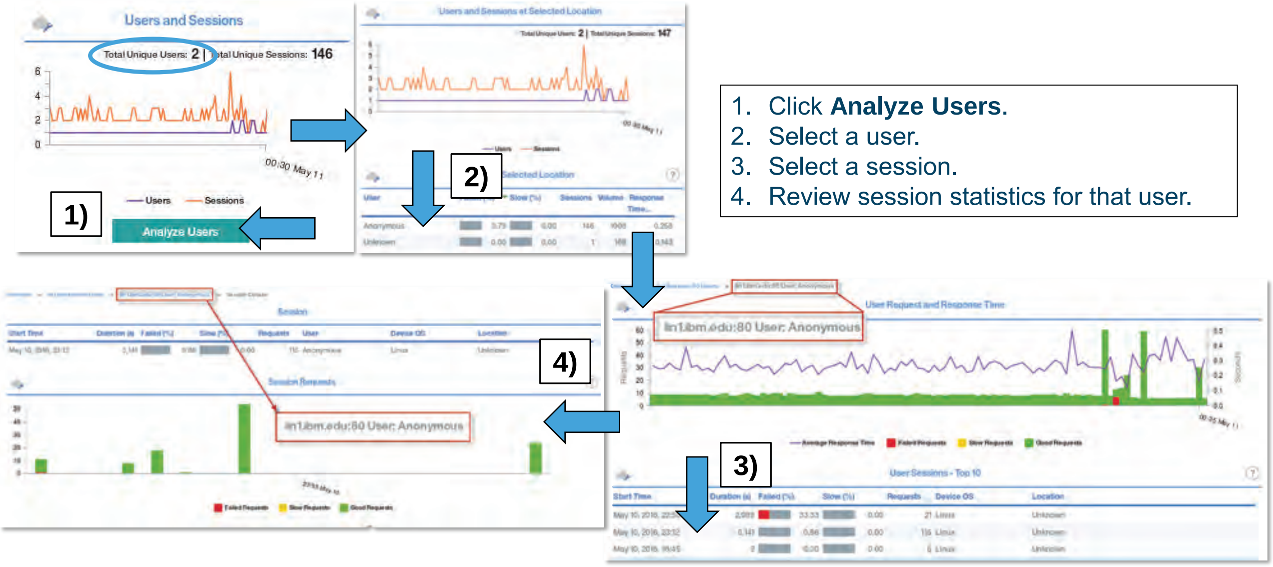
| Transaction | Failed (%) | Slow (%) | Volume | Response Time... |
|---|---|---|---|---|
| /simpletrade/BuyStock<br>/simpletrade/BuyStock | 0.00 | 12.31 | 1,560 | 4,610 |

- Users and Sessions
- Requests by status
- Top 10 worst transactions
- Top five worst user performance
- Top five worst device performance

*User transactions*

Version 8.1.3 presents user-response data differently, and with additional detail. See the next slide for details.

# Viewing user activity

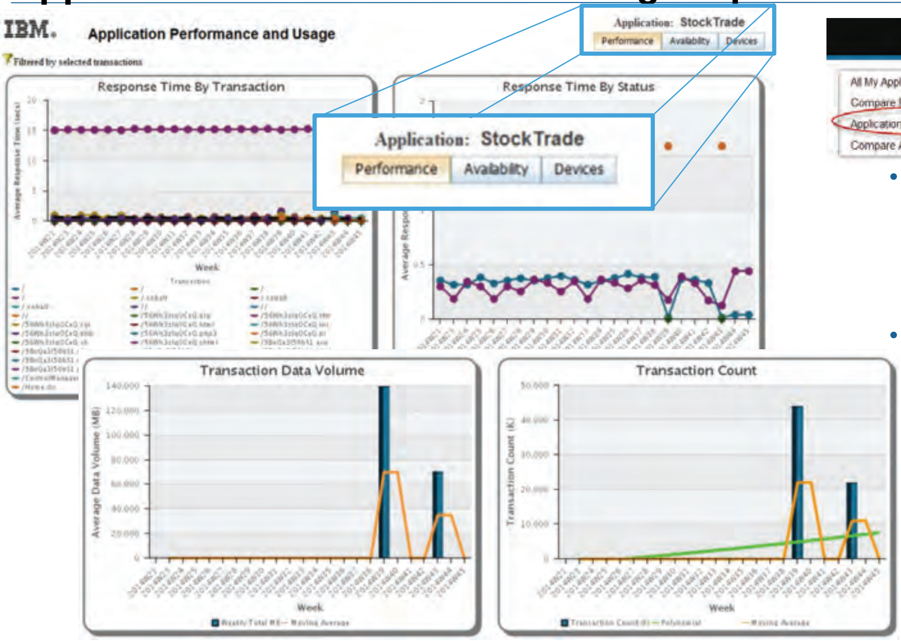

Users and Sessions
Total Unique Users: 2 | Total Unique Sessions: 146
Users
Sessions
Analyze Users
1)
Users and Sessions at Selected Location
Total Unique Users: 2 | Total Unique Sessions: 147
Selected Location
Anonymous
Unknown
2)
lin1.ibm.edu:80 User: Anonymous
User Request and Response Time
Average Response Time
Failed Requests
Slow Requests
Good Requests
User Sessions - Top 10
3)
Session
Session Requests
lin1.ibm.edu:80 User: Anonymous
4)

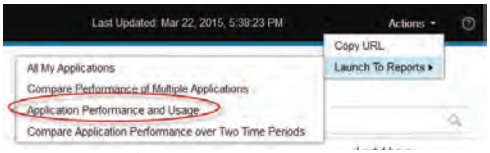
1. Click **Analyze Users**.
2. Select a user.
3. Select a session.
4. Review session statistics for that user.

*Viewing user activity*

The Users and Sessions widget lists those statistics for the current data collection period. Individual users can drill down to the session level.

## Application Performance and Usage reports




Last Updated: Mar 22, 2015, 5:38:23 PM
Actions
Copy URL
Launch To Reports ▸
All My Applications
Compare Performance of Multiple Applications
Application Performance and Usage
Compare Application Performance over Two Time Periods

- Performance views
  - Response Time By Transaction
  - Response Time By Status
  - Transaction Data Volume
  - Transaction Count
- Availability views
  - Successful versus Failed Transaction Percentage
  - Successful versus Failed Transactions By Device
  - Top HTTP Error Codes

Devices views
  - Transactions By Devices
  - Transactions By Device OS
  - Transactions By Device Browser

*Application Performance and Usage reports*

Several application and performance reports are available. This slide shows Application Performance and Usage reports for the time interval that you are currently viewing in the Application Performance Dashboard:

- Response Time By Transaction: Line chart of average response time by (key) transactions
- Response Time By Status: Line chart of average transaction response time by success, server error, and client error
- Transaction Data Volume: Bar and line chart of transaction data volume with transaction data volume average
- Transaction Count: Bar and line charts of transaction count and polynomial values and moving average

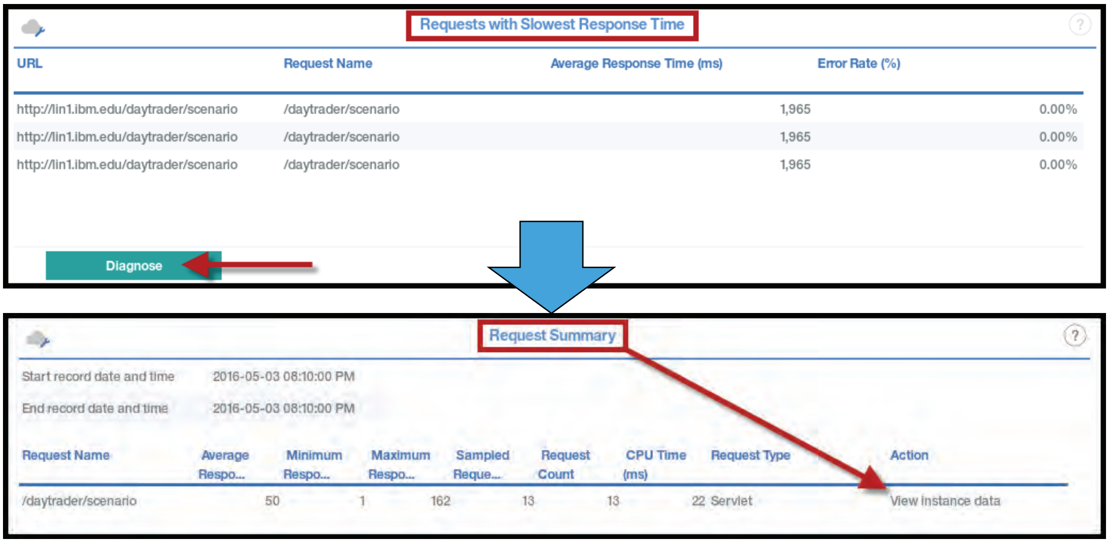
Switch to the **Availability** tab (see inset) to view the following information by selected time interval for the application:

- Stacked bar chart of successful versus failed transaction percentage, with failures as server error and client error
- Stacked bar chart of successful versus failed transaction count by device types
- Pie chart of top error codes

Switch to the **Devices** tab (see inset) to view the following information by selected time interval for the application:

- Bar chart of transactions by device type
- Bar chart of transactions by device operating system
- Bar chart of transaction by device browser
- Table showing transaction performance by dimensions, which you can filter based on device type, device operating system, device brand, and device browser

## Diagnostic mode and method trace (AD)



Requests with Slowest Response Time

| URL | Request Name | Average Response Time (ms) | Error Rate (%) |
|---|---|---|---|
| http://lin1.ibm.edu/daytrader/scenario | /daytrader/scenario | 1,965 | 0.00% |
| http://lin1.ibm.edu/daytrader/scenario | /daytrader/scenario | 1,965 | 0.00% |
| http://lin1.ibm.edu/daytrader/scenario | /daytrader/scenario | 1,965 | 0.00% |

Request Summary

Start record date and time 2016-05-03 08:10:00 PM

End record date and time 2016-05-03 08:10:00 PM

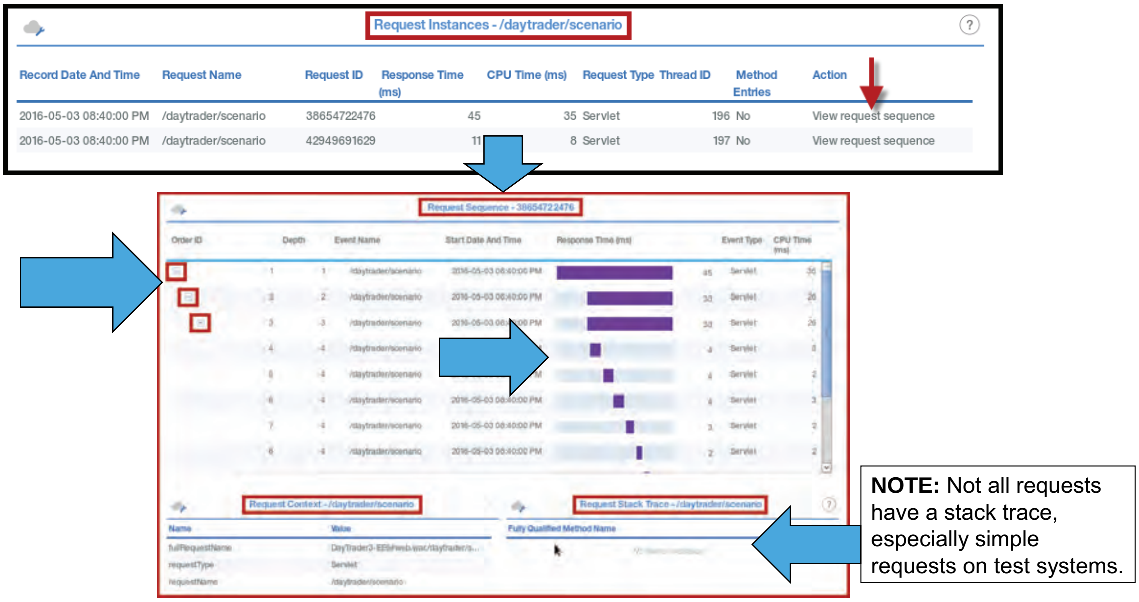
| Request Name | Average Respo... | Minimum Respo... | Maximum Respo... | Sampled Reque... | Request Count | CPU Time (ms) | Request Type | Action |
|---|---|---|---|---|---|---|---|---|
| /daytrader/scenario | 50 | 1 | 162 | 13 | 13 | 22 | Servlet | View instance data |

*Diagnostic mode and method trace (AD)*

When diagnostic mode is enabled, a Diagnose option is available for slow requests on the Requests with Slowest Response Time widget of the Status Overview dashboard.

Clicking **Diagnose** links to a Request Summary table with Request instances. To drill down, click View instance data in the Action column.

## Diagnostic mode and method trace (AD) (Cont.)



Request Instances - /daytrader/scenario

| Record Date And Time | Request Name | Request ID | Response Time (ms) | CPU Time (ms) | Request Type | Thread ID | Method Entries | Action |
|---|---|---|---|---|---|---|---|---|
| 2016-05-03 08:40:00 PM | /daytrader/scenario | 38654722476 | 45 | 35 | Servlet | 196 | No | View request sequence |
| 2016-05-03 08:40:00 PM | /daytrader/scenario | 42949691629 | 11 | 8 | Servlet | 197 | No | View request sequence |

Request Sequence - 38654722476

| Order ID | Depth | Event Name | Start Date And Time | Response Time (ms) | | Event Type | CPU Time (ms) |
|---|---|---|---|---|---|---|---|
| 1 | 1 | /daytrader/scenario | 2016-05-03 08:40:00 PM | | 45 | Servlet | 35 |
| 2 | 2 | /daytrader/scenario | 2016-05-03 08:40:00 PM | | 33 | Servlet | 26 |
| 3 | 3 | /daytrader/scenario | 2016-05-03 08:40:00 PM | | 33 | Servlet | 26 |
| 4 | 4 | /daytrader/scenario | | | 4 | Servlet | 0 |
| 5 | 4 | /daytrader/scenario | | | 4 | Servlet | 2 |
| 6 | 4 | /daytrader/scenario | 2016-05-03 08:40:00 PM | | 4 | Servlet | 3 |
| 7 | 4 | /daytrader/scenario | 2016-05-03 08:40:00 PM | | 3 | Servlet | 2 |
| 8 | 4 | /daytrader/scenario | 2016-05-03 08:40:00 PM | | 2 | Servlet | 2 |

Request Context - /daytrader/scenario

| Name | Value |
|---|---|
| fullRequestName | DayTrader3-EE6/web.war/daytrader/s... |
| requestType | Servlet |
| requestName | /daytrader/scenario |

Request Stack Trace - /daytrader/scenario

Fully Qualified Method Name

**NOTE:** Not all requests have a stack trace, especially simple requests on test systems.

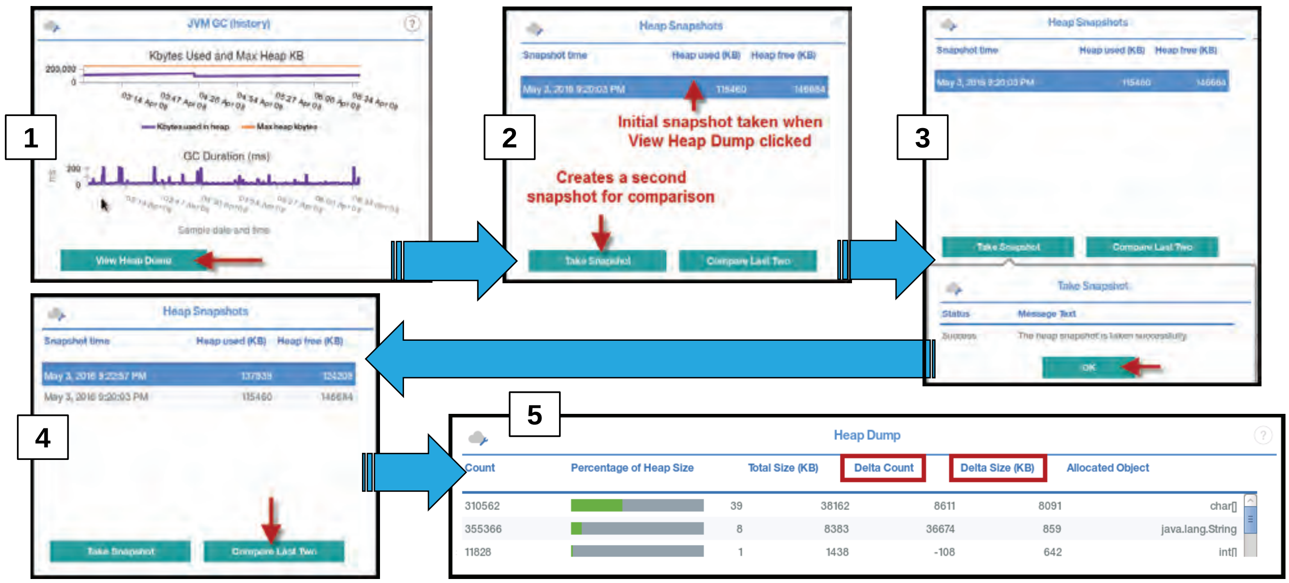
© Copyright IBM Corporation 2016

20

*Diagnostic mode and method trace (AD) (Cont.)*

On the Request Instances widget, locate the instance you want to examine, and click **View request sequence** in the Action column. The request sequence shows the order of events making up the request. Click the plus signs to expand the request. Click individual requests in the sequence to view associated request context and request stack trace.

## Heap memory dump



Initial snapshot taken when View Heap Dump clicked

Creates a second snapshot for comparison

| Status | Message Text |
|---|---|
| Success | The heap snapshot is taken successfully |

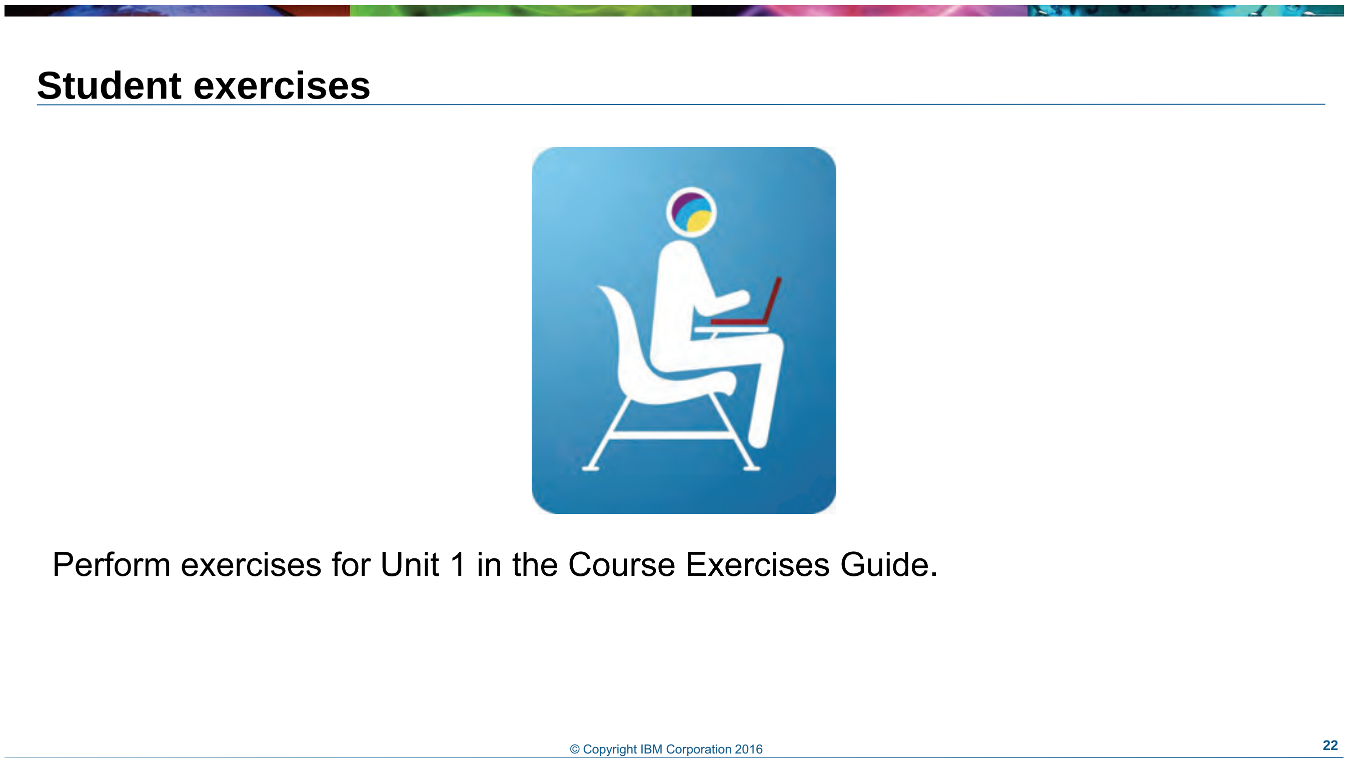
| Snapshot time | Heap used (KB) | Heap free (KB) |
|---|---|---|
| May 3, 2016 9:22:57 PM | 137839 | 124309 |
| May 3, 2016 9:20:03 PM | 115460 | 146684 |

| Count | Percentage of Heap Size | Total Size (KB) | Delta Count | Delta Size (KB) | Allocated Object |
|---|---|---|---|---|---|
| 310562 | 39 | 38162 | 8611 | 8091 | char[] |
| 355366 | 8 | 8383 | 36674 | 859 | java.lang.String |
| 11828 | 1 | 1438 | -108 | 642 | int[] |

*Heap memory dump*

You can perform these tasks with the Heap Dump feature:

- View what is currently on the heap (see #1 in the slide)
- Take a snapshot of the heap (see #2 and #3 in the slide)
- Compare two heap snapshots (see #4 and #5 in the slide)
- View a single heap snapshot (not shown)

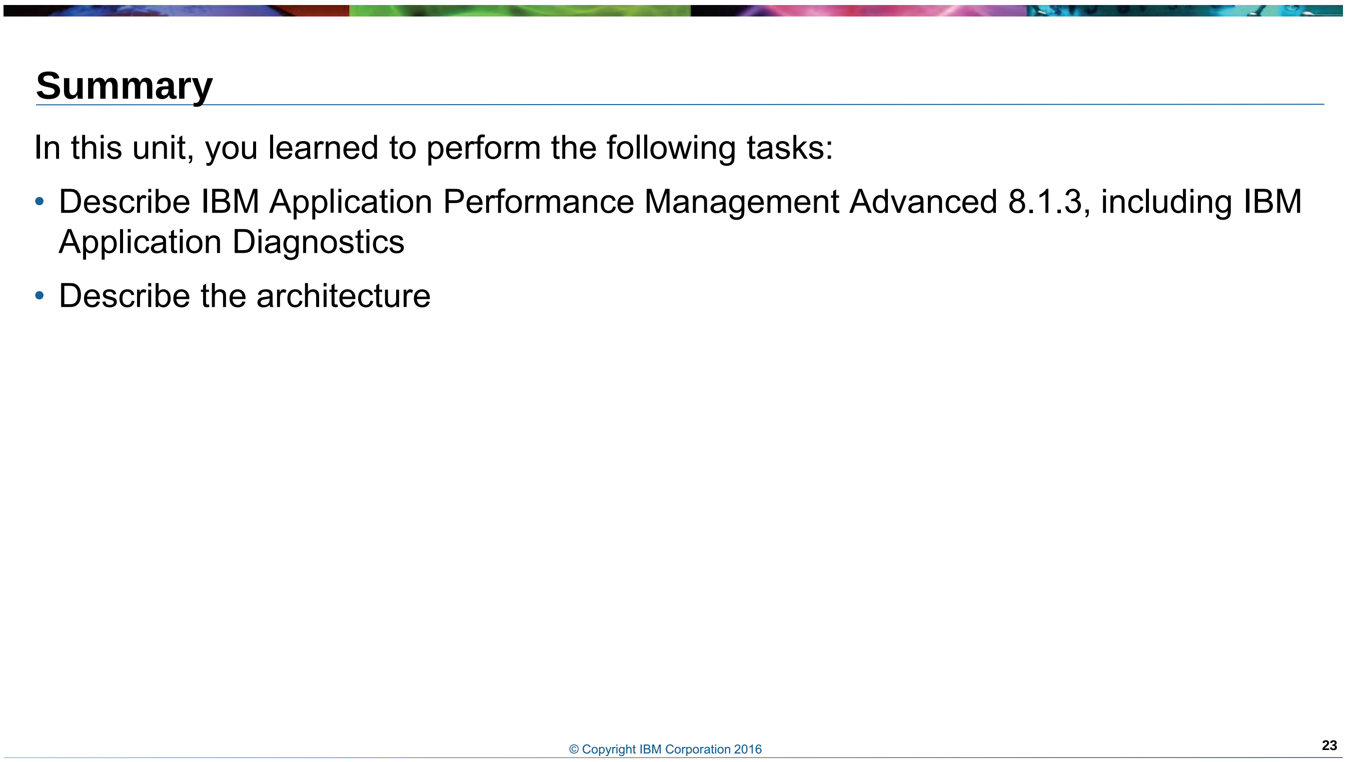


## Student exercises

Perform exercises for Unit 1 in the Course Exercises Guide.

*Student exercises*

Perform exercises for Unit 1 in the Course Exercises Guide.



## Summary

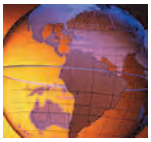
In this unit, you learned to perform the following tasks:

- Describe IBM Application Performance Management Advanced 8.1.3, including IBM Application Diagnostics
- Describe the architecture

*Summary*

In this unit, you learned to perform the following tasks:

- Describe IBM Application Performance Management Advanced 8.1.3, including IBM Application Diagnostics
- Describe the architecture



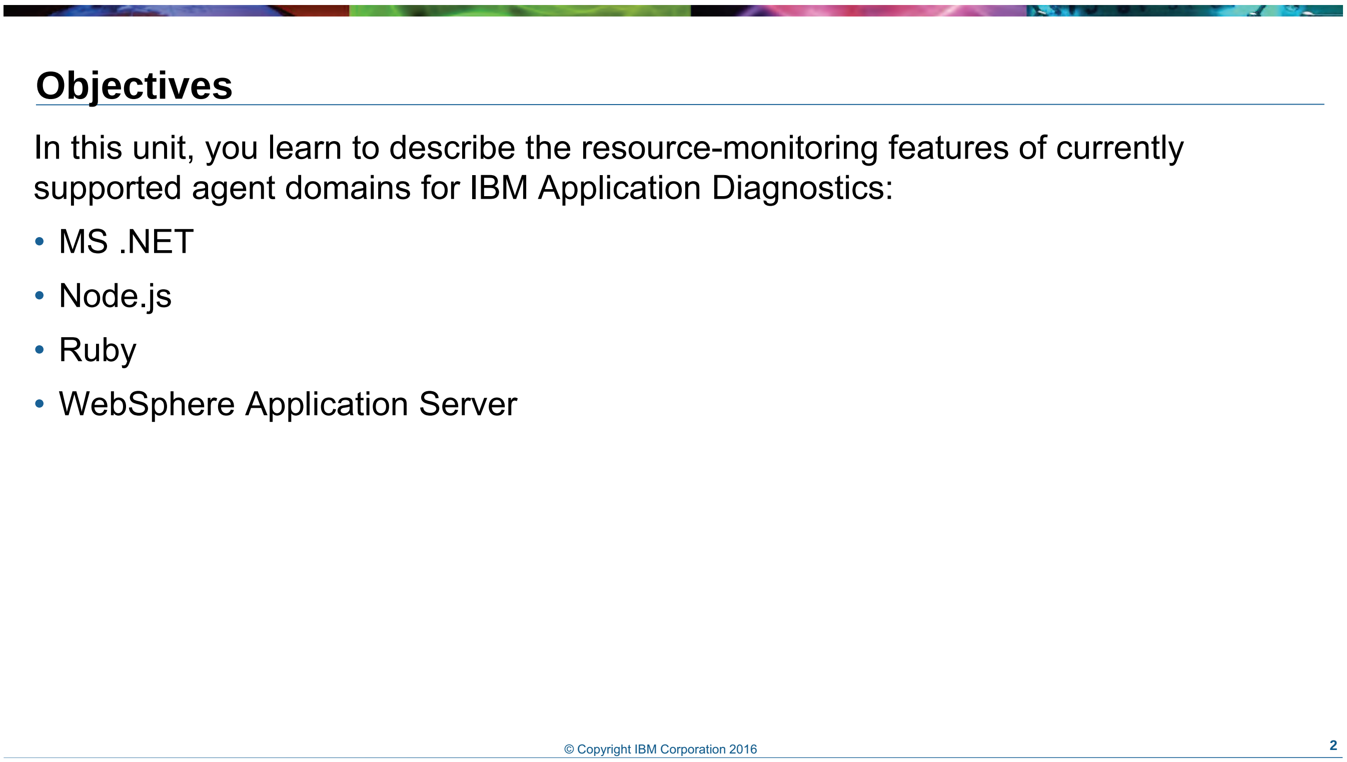
# 2 Monitoring application resources

## 2 Monitoring application resources

© Copyright IBM Corporation 2016

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This presentation is a technical overview of the resource-monitoring features of IBM Application Diagnostics agents.



## Objectives

In this unit, you learn to describe the resource-monitoring features of currently supported agent domains for IBM Application Diagnostics:

- MS .NET
- Node.js
- Ruby
- WebSphere Application Server

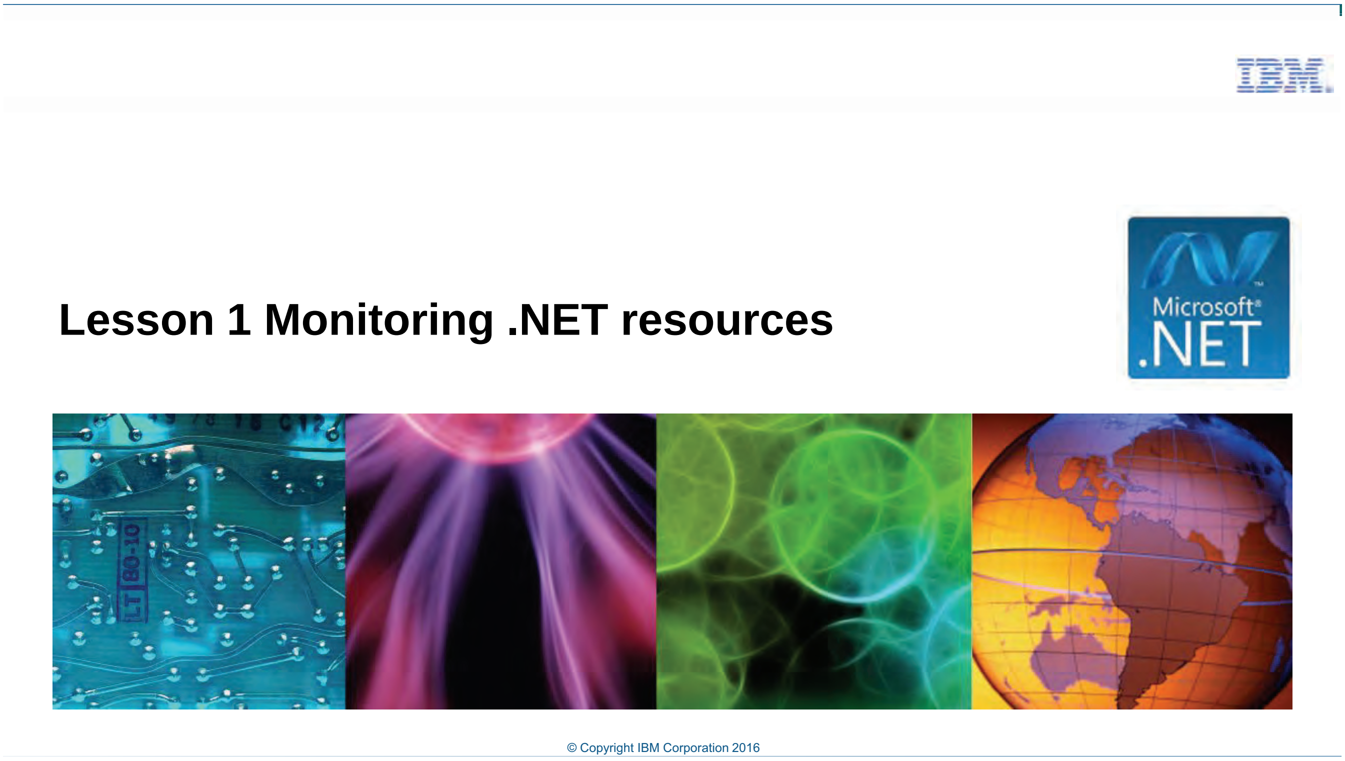
© Copyright IBM Corporation 2016

2

*Objectives*

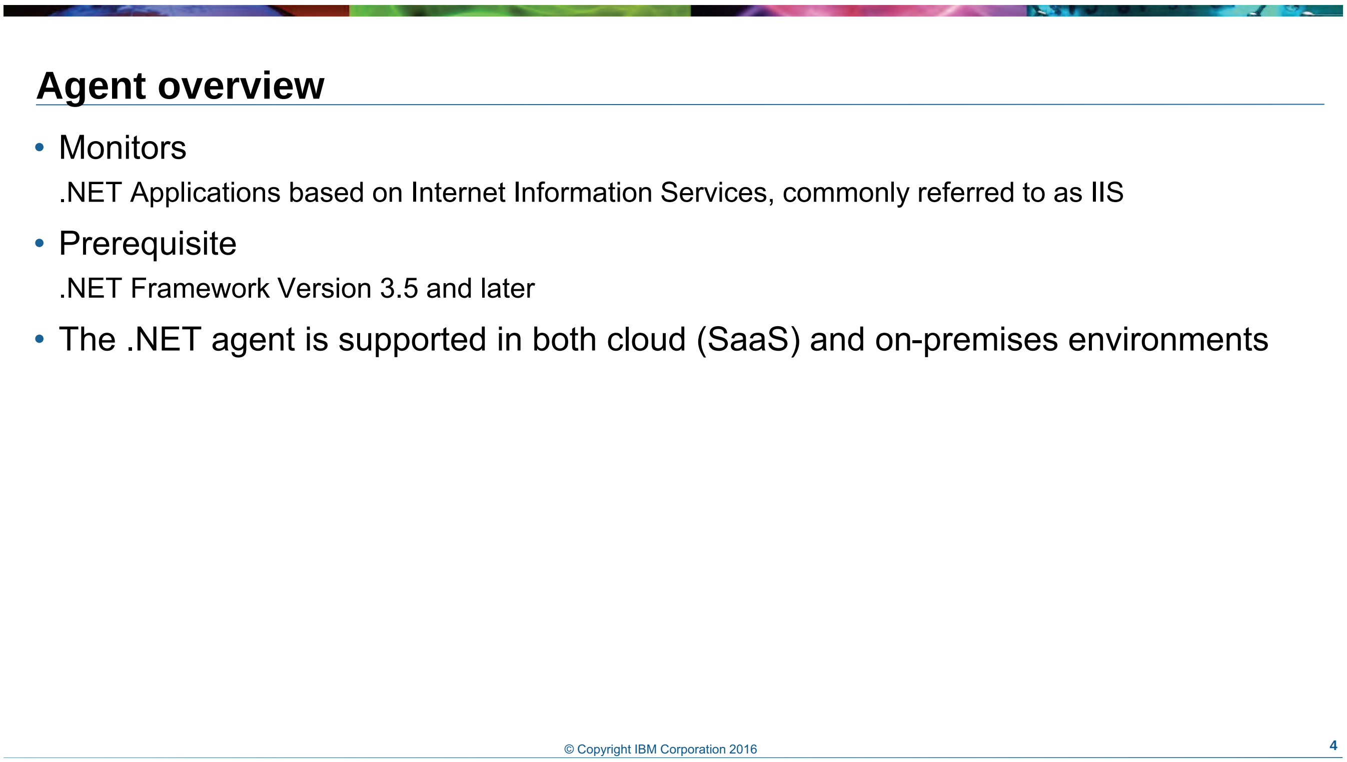
This presentation presents an overview of the resource-monitoring offered by the currently supported agents of IBM Application Diagnostics.

# Lesson 1  Monitoring .NET resources



**Lesson 1 Monitoring .NET resources**

In this section of the presentation, you get a description of key features of the .NET monitoring agent included with IBM Application Diagnostics.



## Agent overview

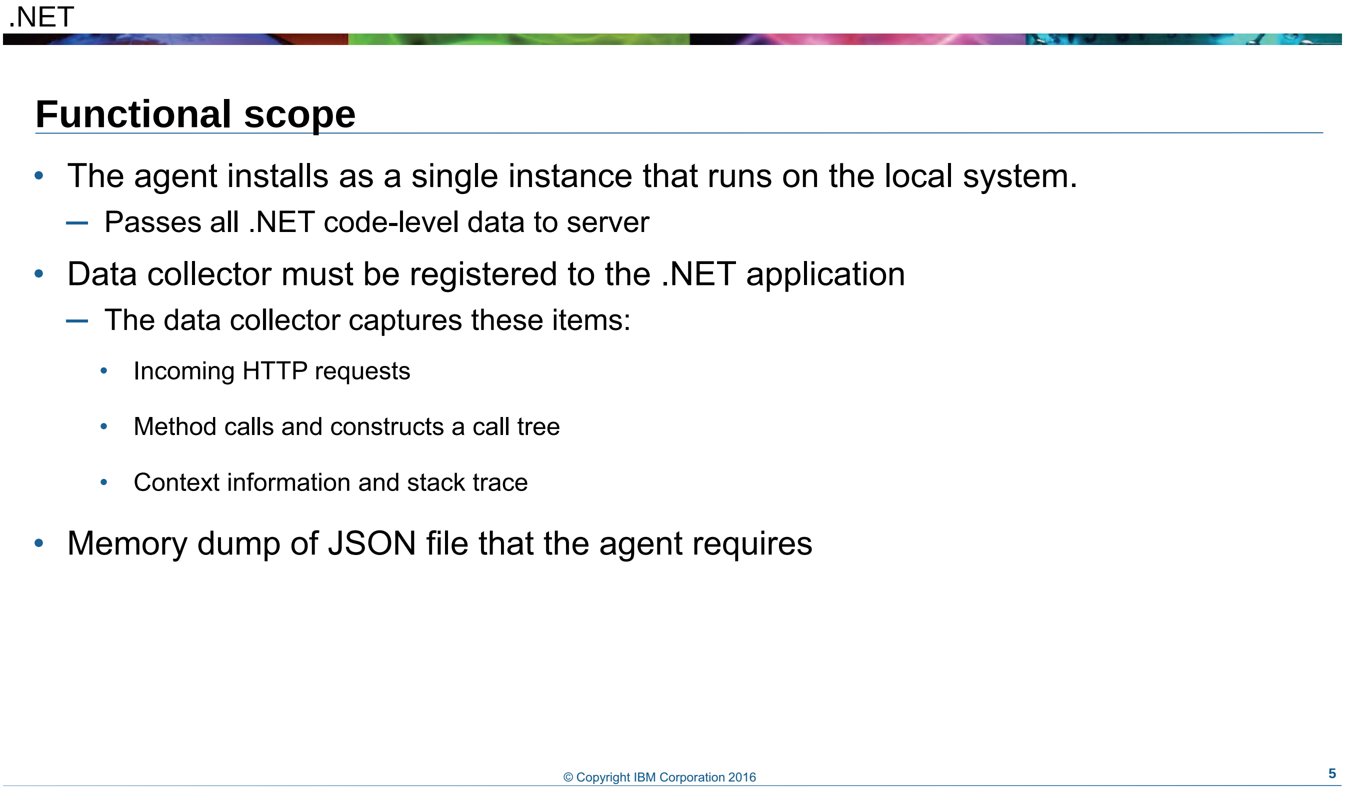
- Monitors

  .NET Applications based on Internet Information Services, commonly referred to as IIS
- Prerequisite

  .NET Framework Version 3.5 and later
- The .NET agent is supported in both cloud (SaaS) and on-premises environments

4

*Agent overview*



.NET

## Functional scope

- The agent installs as a single instance that runs on the local system.
  - Passes all .NET code-level data to server
- Data collector must be registered to the .NET application
  - The data collector captures these items:
    - Incoming HTTP requests
    - Method calls and constructs a call tree
    - Context information and stack trace
- Memory dump of JSON file that the agent requires

5

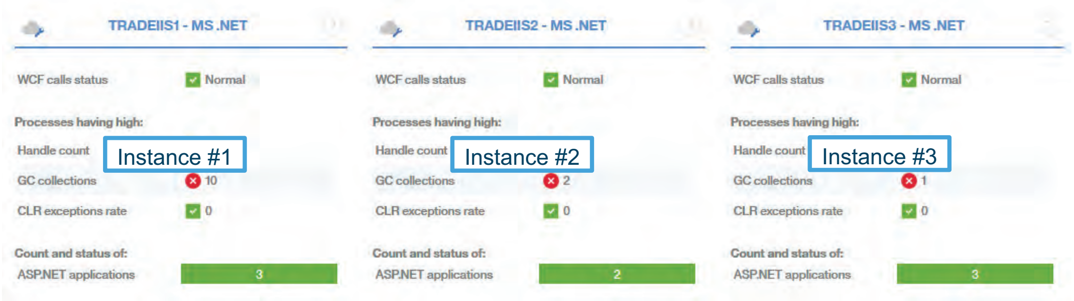
*Functional scope*

The agent installs as a single instance running on the local system. All .NET deep-dive data is passed to the Monitoring Infrastructure Node (MIN). The data collector must be registered to the .NET application.

The data collector captures these items:

- Incoming HTTP requests
- Method calls and constructs a call tree
- Context info and stack trace
- Dump of JSON file required by agent

# .NET status overview



TRADEIIS1 - MS .NET

WCF calls status ✓ Normal

Processes having high:

Handle count — Instance #1

GC collections ☒ 10

CLR exceptions rate ✓ 0

Count and status of:

ASP.NET applications 3

TRADEIIS2 - MS .NET

WCF calls status ✓ Normal

Processes having high:

Handle count — Instance #2

GC collections ☒ 2

CLR exceptions rate ✓ 0

Count and status of:

ASP.NET applications 2

TRADEIIS3 - MS .NET

WCF calls status ✓ Normal

Processes having high:

Handle count — Instance #3

GC collections ☒ 1

CLR exceptions rate ✓ 0

Count and status of:

ASP.NET applications 3

**WCF calls status**
Indicates whether the number of calls that failed per second for services is greater than normal.

**Processes with high:**

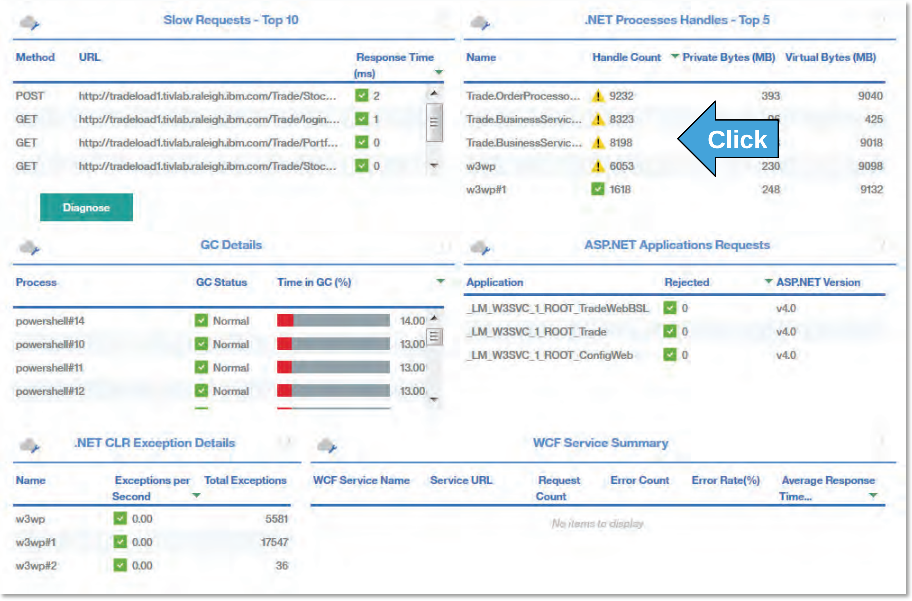
- **Handle count**
  The number of processes with the handle count greater than the threshold.
- **GC collections**
  The number of processes for which the garbage collection status is critical or time for garbage collection is greater than normal.
- **ASP.NET applications (critical)**
  The number of ASP.NET applications with high request rejections.
- **ASP.NET applications (normal)**
  The count of ASP.NET applications with no request rejections.
- **CLR exceptions rate**
  The number of processes for which exceptions per second are greater than normal.

*.NET status overview*

The Status Overview contains one widget per .NET instance. The widget shows high-level status and resource data for .NET KPIs.

Clicking an instance widget gives you access to a dashboard with clickable widgets for .NET resources.

## .NET resource dashboard



**Slow Requests - Top 10**

| Method | URL | Response Time (ms) |
|---|---|---|
| POST | http://tradeload1.tivlab.raleigh.ibm.com/Trade/Stoc... | ✓ 2 |
| GET | http://tradeload1.tivlab.raleigh.ibm.com/Trade/login.... | ✓ 1 |
| GET | http://tradeload1.tivlab.raleigh.ibm.com/Trade/Portf... | ✓ 0 |
| GET | http://tradeload1.tivlab.raleigh.ibm.com/Trade/Stoc... | ✓ 0 |

**.NET Processes Handles - Top 5**

| Name | Handle Count | Private Bytes (MB) | Virtual Bytes (MB) |
|---|---|---|---|
| Trade.OrderProcesso... | ⚠ 9232 | 393 | 9040 |
| Trade.BusinessServic... | ⚠ 8323 | 96 | 425 |
| Trade.BusinessServic... | ⚠ 8198 | [illegible] | 9018 |
| w3wp | ⚠ 6053 | 230 | 9098 |
| w3wp#1 | ✓ 1618 | 248 | 9132 |

**GC Details**

| Process | GC Status | Time in GC (%) |
|---|---|---|
| powershell#14 | ✓ Normal | 14.00 |
| powershell#10 | ✓ Normal | 13.00 |
| powershell#11 | ✓ Normal | 13.00 |
| powershell#12 | ✓ Normal | 13.00 |

**ASP.NET Applications Requests**

| Application | Rejected | ASP.NET Version |
|---|---|---|
| _LM_W3SVC_1_ROOT_TradeWebBSL | ✓ 0 | v4.0 |
| _LM_W3SVC_1_ROOT_Trade | ✓ 0 | v4.0 |
| _LM_W3SVC_1_ROOT_ConfigWeb | ✓ 0 | v4.0 |

**.NET CLR Exception Details**

| Name | Exceptions per Second | Total Exceptions |
|---|---|---|
| w3wp | ✓ 0.00 | 5581 |
| w3wp#1 | ✓ 0.00 | 17547 |
| w3wp#2 | ✓ 0.00 | 36 |

**WCF Service Summary**

| WCF Service Name | Service URL | Request Count | Error Count | Error Rate(%) | Average Response Time... |
|---|---|---|---|---|---|

*No items to display*

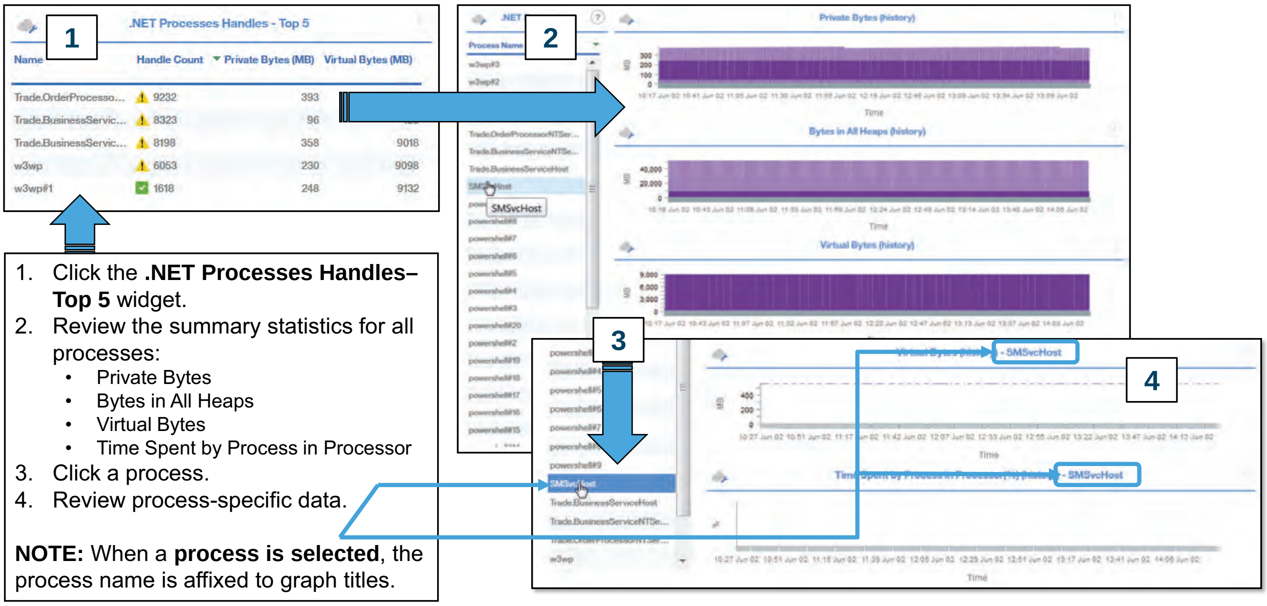
*.NET resource dashboard*

The main dashboard for the .NET agent offers resource usage information, as well as a summary of requests and their relative performance.

In all widgets except .NET CLR Exception Details, you can drill down to additional dashboards. See the following slides for widget-specific information.

The Request Summary widget is covered in Unit : .

## .NET processes



.NET Processes Handles - Top 5

| Name | Handle Count | Private Bytes (MB) | Virtual Bytes (MB) |
|---|---|---|---|
| Trade.OrderProcesso... | ⚠ 9232 | 393 | |
| Trade.BusinessServic... | ⚠ 8323 | 96 | |
| Trade.BusinessServic... | ⚠ 8198 | 358 | 9018 |
| w3wp | ⚠ 6053 | 230 | 9098 |
| w3wp#1 | ✓ 1618 | 248 | 9132 |

1
2
3
4
Process Name
Private Bytes (history)
Bytes in All Heaps (history)
Virtual Bytes (history)
Time
MB
SMSvcHost
Virtual Bytes (history) - SMSvcHost
Time Spent by Process in Processor(%) (history) - SMSvcHost

1. Click the **.NET Processes Handles–Top 5** widget.
2. Review the summary statistics for all processes:
   - Private Bytes
   - Bytes in All Heaps
   - Virtual Bytes
   - Time Spent by Process in Processor
3. Click a process.
4. Review process-specific data.

**NOTE:** When a **process is selected**, the process name is affixed to graph titles.

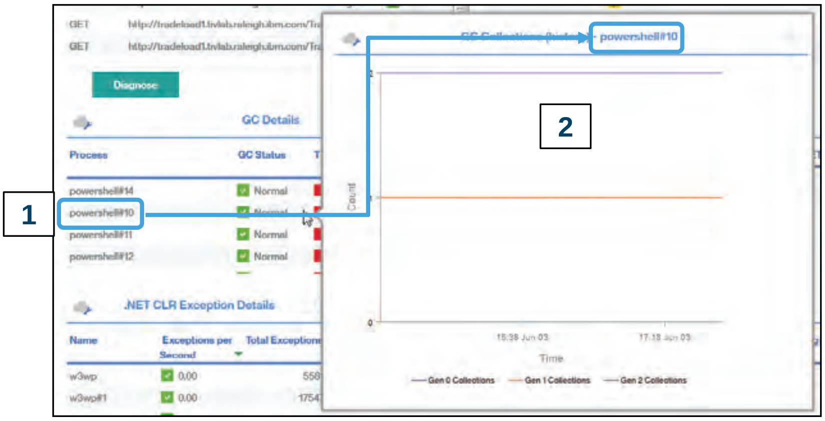
*.NET processes*

Click the .NET Processes Handles – Top 5 widget to bring up a dashboard with summary graphs for all processes. Click a specific process to view process-specific graphs.

# Garbage collection

1. Click a row in the **GC Status** widget to view a window for the selected garbage collection process.
2. Review garbage collection details for the process.

**NOTE:** When a **process is selected**, the process name is affixed to the graph title.



GET http://tradeload1.tivlab.raleigh.ibm.com/Tr
GET http://tradeload1.tivlab.raleigh.ibm.com/Tr

Diagnose

GC Details

| Process | GC Status |
|---|---|
| powershell#14 | ✓ Normal |
| powershell#10 | ✓ Normal |
| powershell#11 | ✓ Normal |
| powershell#12 | ✓ Normal |

.NET CLR Exception Details

| Name | Exceptions per Second | Total Exceptions |
|---|---|---|
| w3wp | ✓ 0.00 | 558 |
| w3wp#1 | ✓ 0.00 | 1754 |

GC Collections (history) - powershell#10

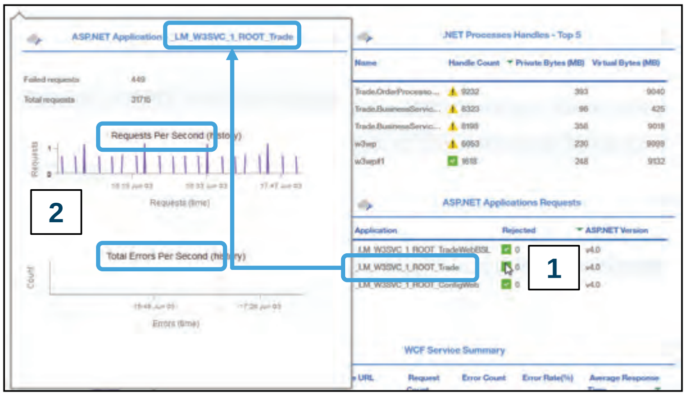
*Garbage collection*

Click the GC Status widget to view a pop-up with process-specific garbage collection data.

## ASP.NET application requests

1. Click a row in the **ASP.NET Application Requests** widget to view a window for the selected application.
2. Review request and error data for the application.

**NOTE:** When an **application is selected**, the name is affixed to graph titles.



ASP.NET Application _LM_W3SVC_1_ROOT_Trade

| | |
|---|---|
| Failed requests | 449 |
| Total requests | 31715 |

Requests Per Second (history)
Requests
1
0
Requests (time)
2
Total Errors Per Second (history)
Count
Errors (time)
1

.NET Processes Handles - Top 5

| Name | Handle Count | Private Bytes (MB) | Virtual Bytes (MB) |
|---|---|---|---|
| Trade.OrderProcesso... | ⚠ 9232 | 393 | 9040 |
| Trade.BusinessServic... | ⚠ 8323 | 96 | 425 |
| Trade.BusinessServic... | ⚠ 8198 | 356 | 9018 |
| w3wp | ⚠ 6053 | 230 | 9099 |
| w3wp#1 | ✓ 1618 | 248 | 9132 |

ASP.NET Applications Requests

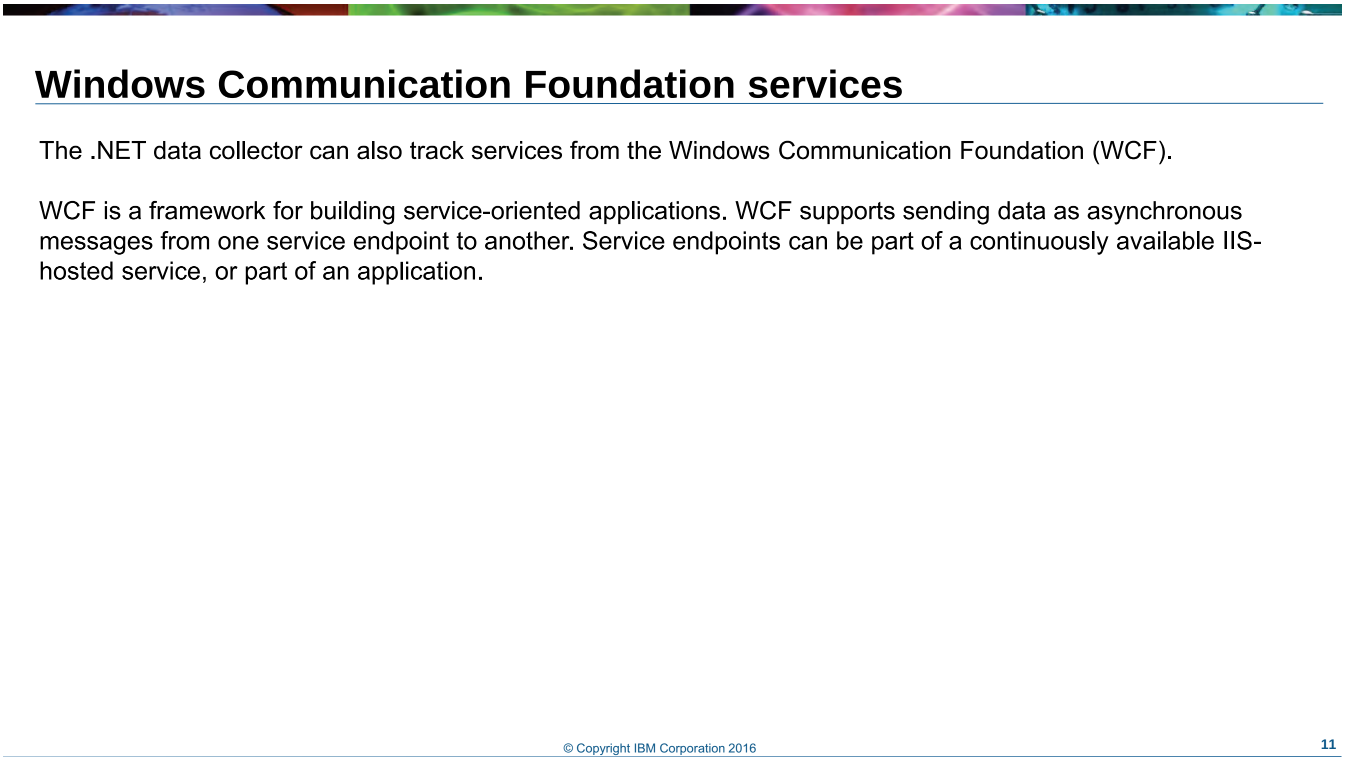
| Application | Rejected | ASP.NET Version |
|---|---|---|
| _LM_W3SVC_1_ROOT_TradeWebBSL | ✓ 0 | v4.0 |
| _LM_W3SVC_1_ROOT_Trade | ✓ 0 | v4.0 |
| _LM_W3SVC_1_ROOT_ConfigWeb | ✓ 0 | v4.0 |

WCF Service Summary

| URL | Request Count | Error Count | Error Rate(%) | Average Response Time |
|---|---|---|---|---|

*ASP.NET application requests*

Click the ASP.NET Application Requests widget to view a pop-up with request and error data. Click a specific application to view application-specific data for requests and errors per second.



## Windows Communication Foundation services

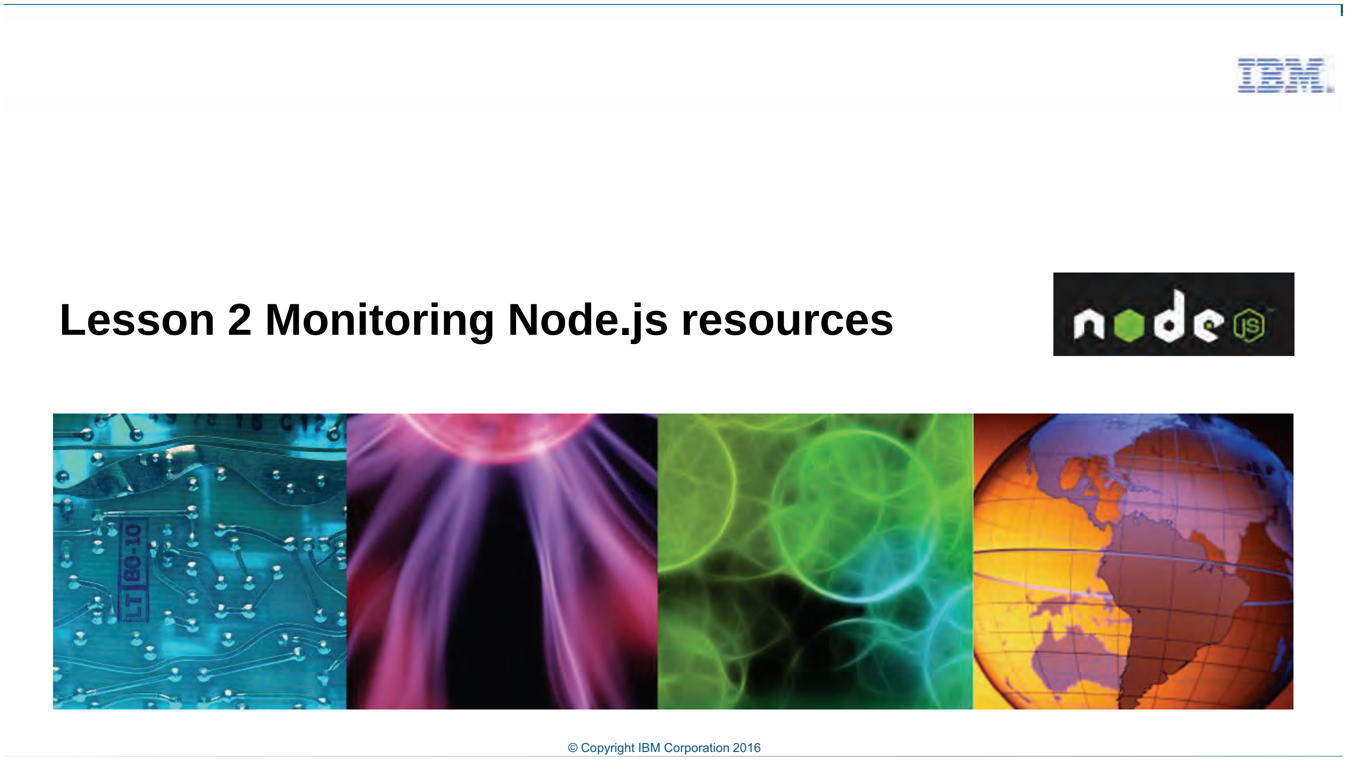
The .NET data collector can also track services from the Windows Communication Foundation (WCF).

WCF is a framework for building service-oriented applications. WCF supports sending data as asynchronous messages from one service endpoint to another. Service endpoints can be part of a continuously available IIS-hosted service, or part of an application.

*Windows Communication Foundation services*

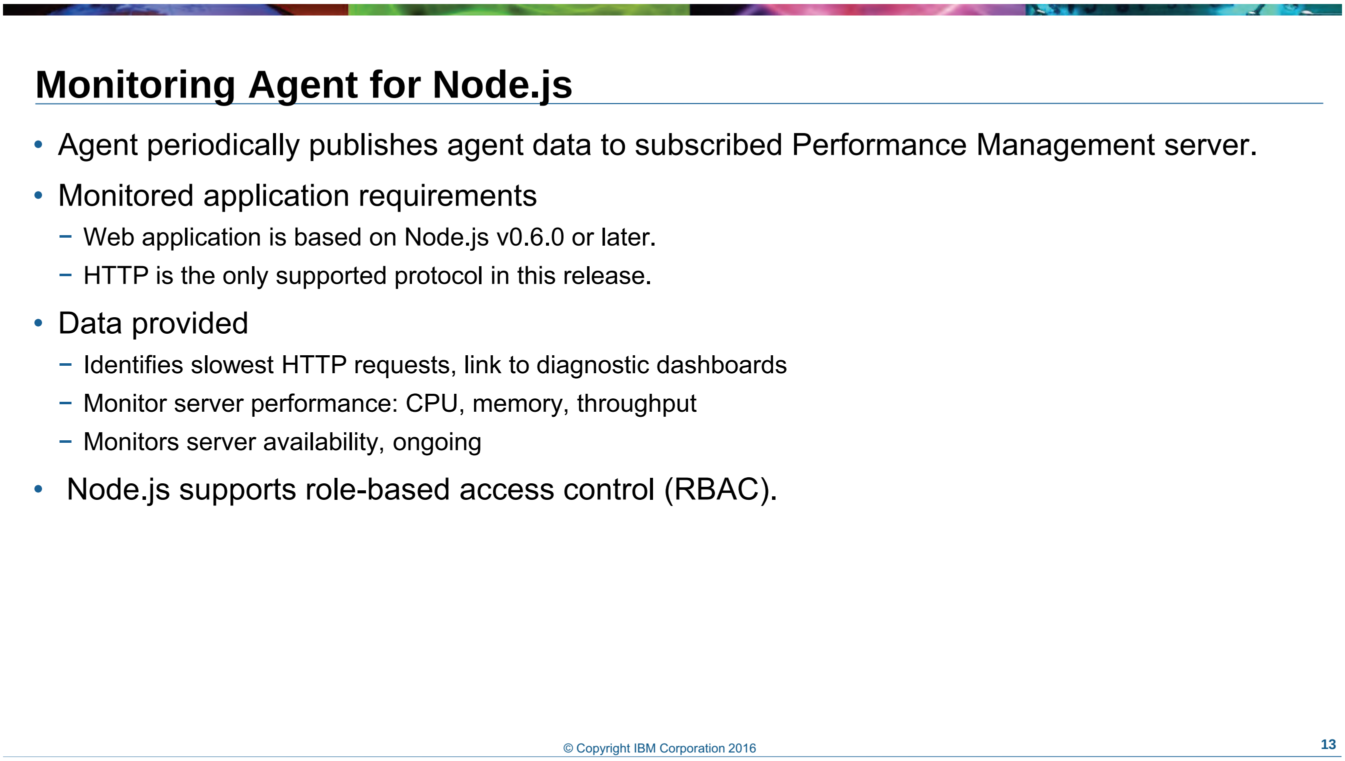
The .NET data collector can also track services from the Windows Communication Foundation (WCF).

# Lesson 2  Monitoring Node.js resources



## Lesson 2 Monitoring Node.js resources

In this section of the presentation, you get a description of key features of the Node.js monitoring agent included with IBM Application Diagnostics.



## Monitoring Agent for Node.js

- Agent periodically publishes agent data to subscribed Performance Management server.
- Monitored application requirements
  - Web application is based on Node.js v0.6.0 or later.
  - HTTP is the only supported protocol in this release.
- Data provided
  - Identifies slowest HTTP requests, link to diagnostic dashboards
  - Monitor server performance: CPU, memory, throughput
  - Monitors server availability, ongoing
- Node.js supports role-based access control (RBAC).

*Monitoring Agent for Node.js*

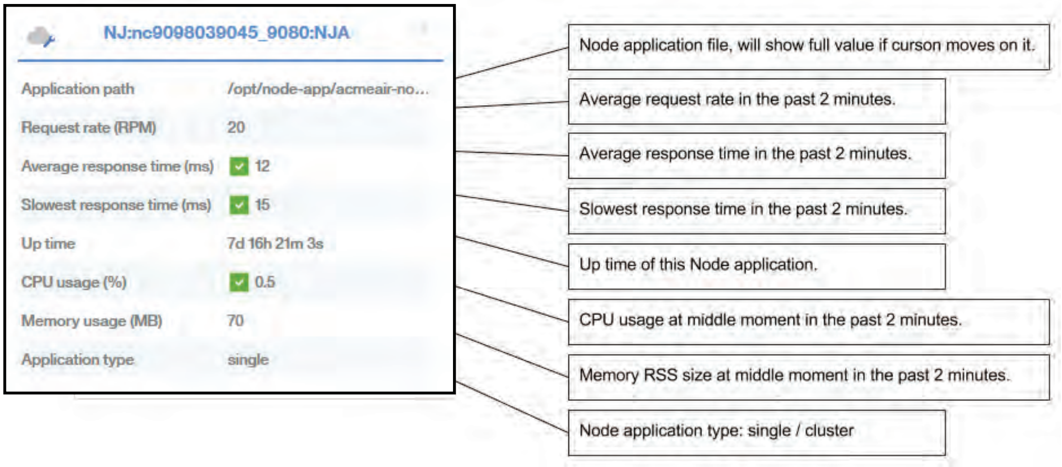
Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, nonblocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

The Monitoring Agent for Node.js periodically publishes agent data to subscribed Performance Management server.

The monitored application must be a web application based on Node.js v0.6.0 or later, using HTTP. The data provided shows the slowest HTTP requests, as well as server performance metrics for CPU, memory, and throughput.

It also monitors server availability.

## Node.js status overview



NJ:nc9098039045_9080:NJA

| | |
|---|---|
| Application path | /opt/node-app/acmeair-no… |
| Request rate (RPM) | 20 |
| Average response time (ms) | ✓ 12 |
| Slowest response time (ms) | ✓ 15 |
| Up time | 7d 16h 21m 3s |
| CPU usage (%) | ✓ 0.5 |
| Memory usage (MB) | 70 |
| Application type | single |

Node application file, will show full value if curson moves on it.

Average request rate in the past 2 minutes.

Average response time in the past 2 minutes.

Slowest response time in the past 2 minutes.

Up time of this Node application.

CPU usage at middle moment in the past 2 minutes.

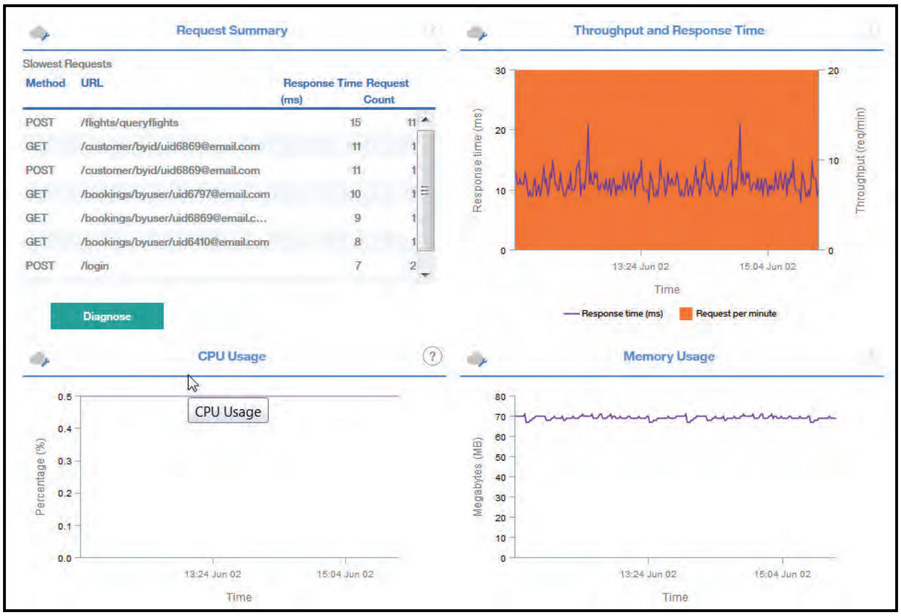
Memory RSS size at middle moment in the past 2 minutes.

Node application type: single / cluster

*Node.js status overview*

The Status Overview contains one widget per Node.js instance. The widget shows high-level status and resource data for Node.js KPIs. Clicking that widget gives you access to a resource dashboard for Node.js.

## Node.js resource dashboard



Request Summary

Slowest Requests

| Method | URL | Response Time (ms) | Request Count |
|---|---|---|---|
| POST | /flights/queryflights | 15 | 11 |
| GET | /customer/byid/uid6869@email.com | 11 | 1 |
| POST | /customer/byid/uid6869@email.com | 11 | 1 |
| GET | /bookings/byuser/uid6797@email.com | 10 | 1 |
| GET | /bookings/byuser/uid6869@email.c... | 9 | 1 |
| GET | /bookings/byuser/uid6410@email.com | 8 | 1 |
| POST | /login | 7 | 2 |

*Node.js resource dashboard*

The main dashboard for the Node.js agent offers resource usage information, as well as a summary of requests and their relative performance.

As of this writing, the three resource widgets provide a single layer of resource data. No further drill-down is supported.

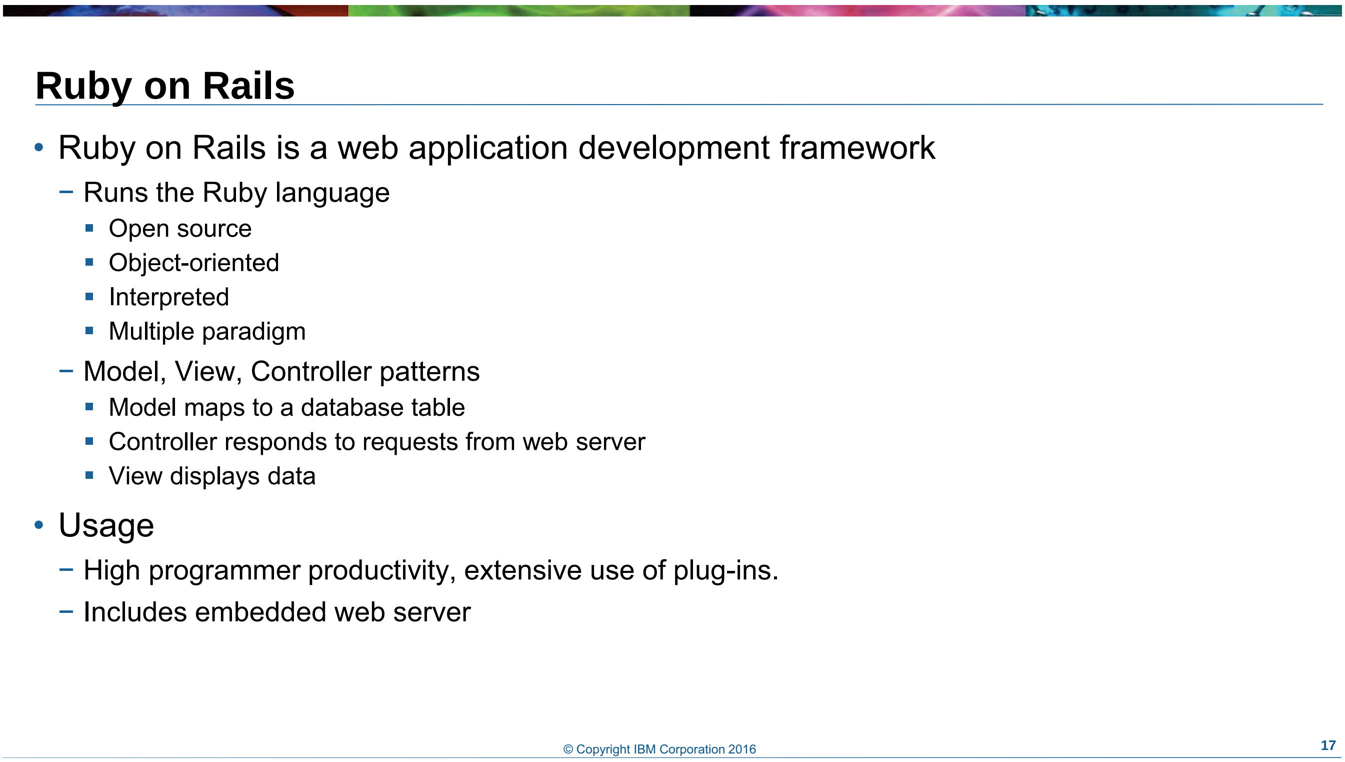
The Request Summary widget is covered in Unit : .

# Lesson 3  Monitoring Ruby resources

Lesson 3 Monitoring Ruby resources

© Copyright IBM Corporation 2016

In this section of the presentation, you get a description of key features of the Ruby monitoring agent included with IBM Application Diagnostics.



## Ruby on Rails

- Ruby on Rails is a web application development framework
  - Runs the Ruby language
    - Open source
    - Object-oriented
    - Interpreted
    - Multiple paradigm
  - Model, View, Controller patterns
    - Model maps to a database table
    - Controller responds to requests from web server
    - View displays data
- Usage
  - High programmer productivity, extensive use of plug-ins.
  - Includes embedded web server

© Copyright IBM Corporation 2016

17

*Ruby on Rails*

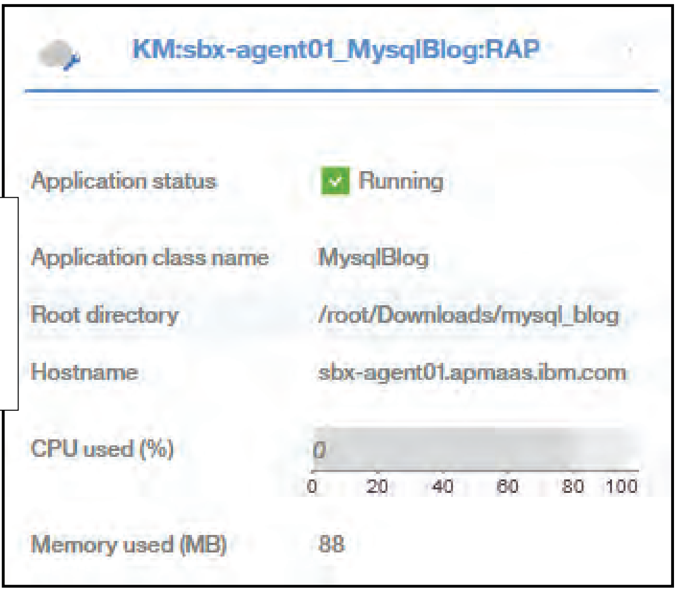
The Monitoring Agent for Ruby monitors applications that run in the Ruby on rails environment. The transactions are written in the Ruby programming language, which shows some similarities to Perl and Smalltalk. Ruby is said to be multi-paradigm because you can mix procedural programming style with object-oriented programming.

Ruby applications use a model, view, controller framework, where the model maps to a table in a database, and a controller is the component that responds to requests for data from the web server. The view is the output, such as the chart or diagram.

Ruby is known for allowing high programmer productivity. Some of this productivity gain is due to the extensive use of plug-ins. Ruby applications are typically not connected directly to the Internet, but operate through a front end web server.

## Ruby status overview

- Application status
- Application class name
- Root directory
- Host name
- CPU% used
- Memory in use (MB)



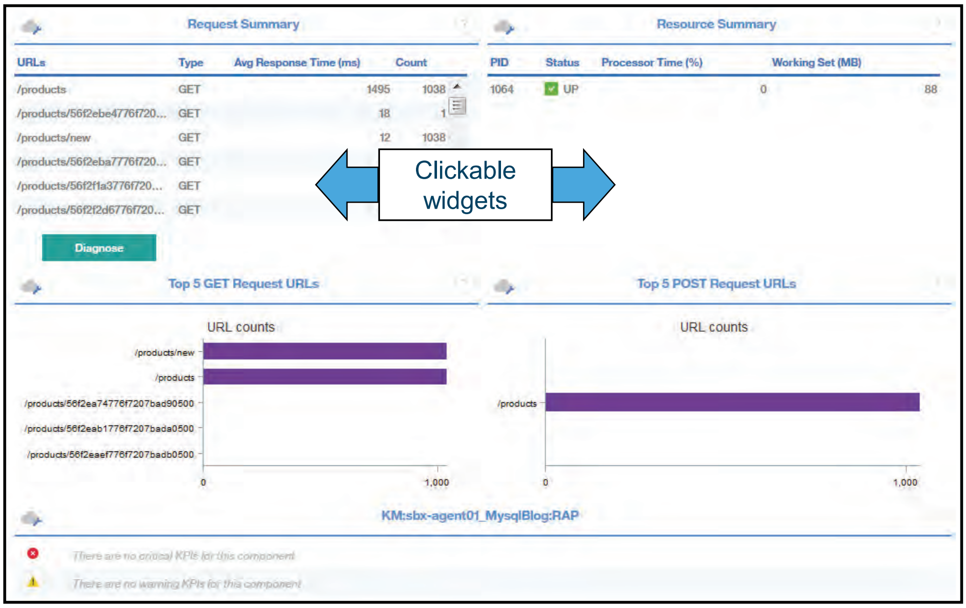
KM:sbx-agent01_MysqlBlog:RAP

| | |
|---|---|
| Application status | ✓ Running |
| Application class name | MysqlBlog |
| Root directory | /root/Downloads/mysql_blog |
| Hostname | sbx-agent01.apmaas.ibm.com |
| CPU used (%) | 0 |
| Memory used (MB) | 88 |

*Ruby status overview*

The Status Overview contains one widget per Ruby instance. The widget shows high-level status and resource data for Ruby KPIs. Clicking that widget gives you access to a dashboard with clickable widgets for Ruby resources.

## Ruby resource dashboard



Request Summary

| URLs | Type | Avg Response Time (ms) | Count |
|---|---|---|---|
| /products | GET | 1495 | 1038 |
| /products/56f2ebe4776f720... | GET | 18 | 1 |
| /products/new | GET | 12 | 1038 |
| /products/56f2eba7776f720... | GET | | |
| /products/56f2f1a3776f720... | GET | | |
| /products/56f2f2d6776f720... | GET | | |

Diagnose

Resource Summary

| PID | Status | Processor Time (%) | Working Set (MB) |
|---|---|---|---|
| 1064 | ✓ UP | 0 | 88 |

Clickable widgets

Top 5 GET Request URLs

URL counts
/products/new
/products
/products/56f2ea74776f7207bad90500
/products/56f2eab1776f7207bada0500
/products/56f2eaef776f7207badb0500
0
1,000

Top 5 POST Request URLs

KM:sbx-agent01_MysqlBlog:RAP

There are no critical KPIs for this component

There are no warning KPIs for this component

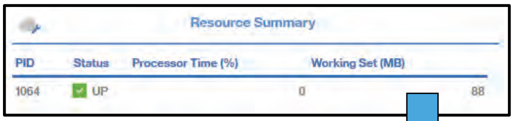
19

*Ruby resource dashboard*

The main dashboard for the Ruby agent offers resource usage information, as well as a summary of requests and their relative performance.

With the Request Summary and Resource Summary widgets, you can drill down to additional dashboards. See the following slide for widget-specific information. The Request Summary widget is covered in Unit : .

## Ruby resource summary

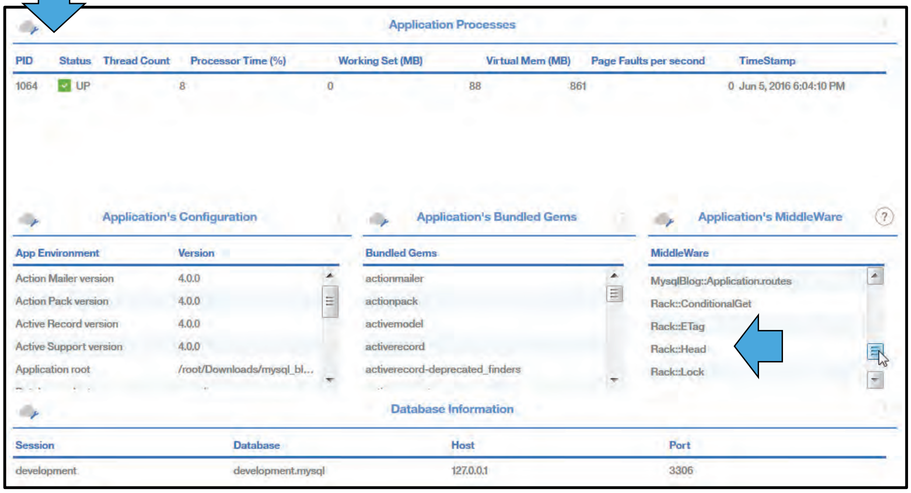


Resource Summary

| PID | Status | Processor Time (%) | Working Set (MB) |
|---|---|---|---|
| 1064 | ✓ UP | 0 | 88 |

Click the **Resource Summary** widget to access application-level data for:

- Processes
- Configuration
- Bundles gems
- Middleware
- Database information



Application Processes

| PID | Status | Thread Count | Processor Time (%) | Working Set (MB) | Virtual Mem (MB) | Page Faults per second | TimeStamp |
|---|---|---|---|---|---|---|---|
| 1064 | ✓ UP | 8 | 0 | 88 | 861 | 0 | Jun 5, 2016 6:04:10 PM |

Application's Configuration

| App Environment | Version |
|---|---|
| Action Mailer version | 4.0.0 |
| Action Pack version | 4.0.0 |
| Active Record version | 4.0.0 |
| Active Support version | 4.0.0 |
| Application root | /root/Downloads/mysql_bl... |

Application's Bundled Gems

| Bundled Gems |
|---|
| actionmailer |
| actionpack |
| activemodel |
| activerecord |
| activerecord-deprecated_finders |

Application's MiddleWare

| MiddleWare |
|---|
| MysqlBlog::Application.routes |
| Rack::ConditionalGet |
| Rack::ETag |
| Rack::Head |
| Rack::Lock |

Database Information

| Session | Database | Host | Port |
|---|---|---|---|
| development | development.mysql | 127.0.0.1 | 3306 |

*Ruby resource summary*

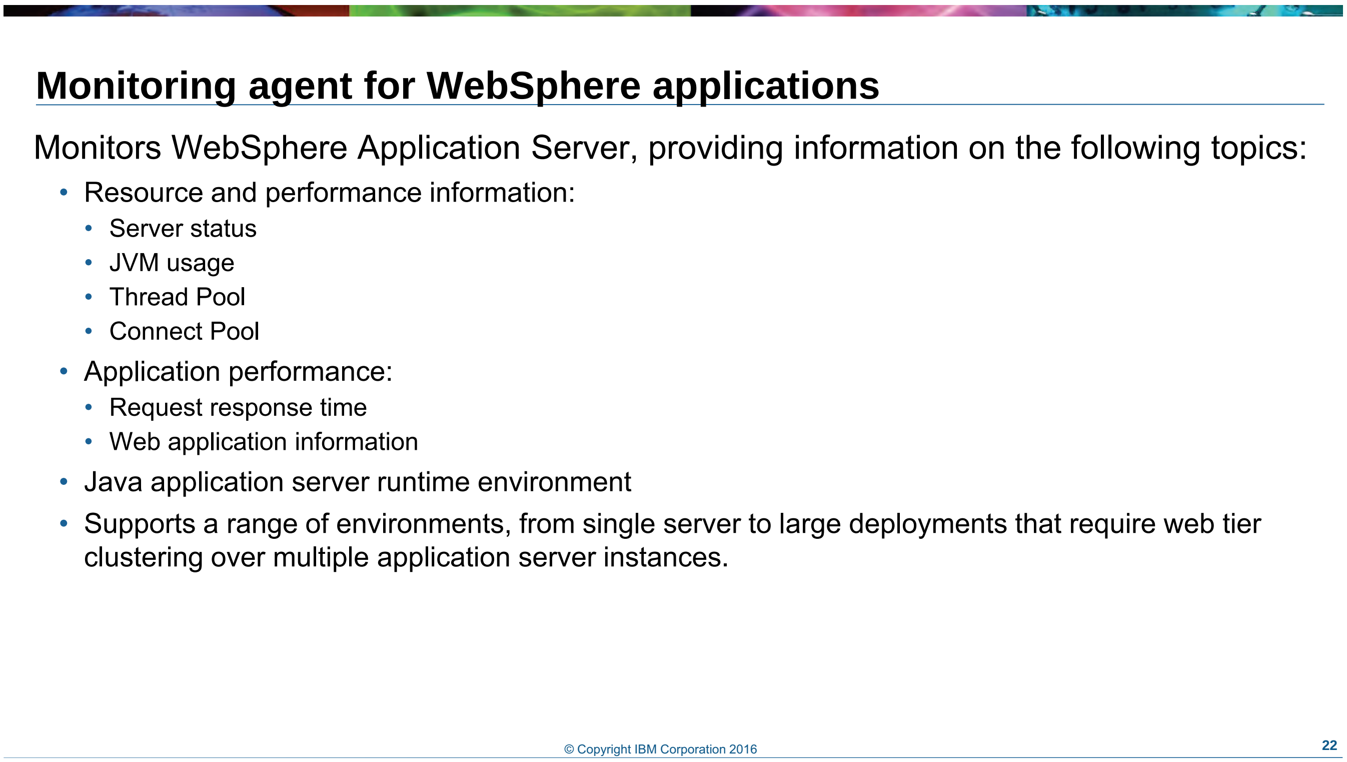
The Resource Summary widget gives you access to a range of application data. Application configuration includes version and directory information.

*Gems* are a distribution mechanism for Ruby programs and libraries. The Middleware widget in this case includes Rack objects. *Rack* is a modular and interface for developing web applications in Ruby.

# Lesson 4  Monitoring WebSphere resources

Lesson 4 Monitoring WebSphere resources

In this section of the presentation, you get a description of key features of the WebSphere Application Server monitoring agent included with IBM Application Diagnostics.



## Monitoring agent for WebSphere applications

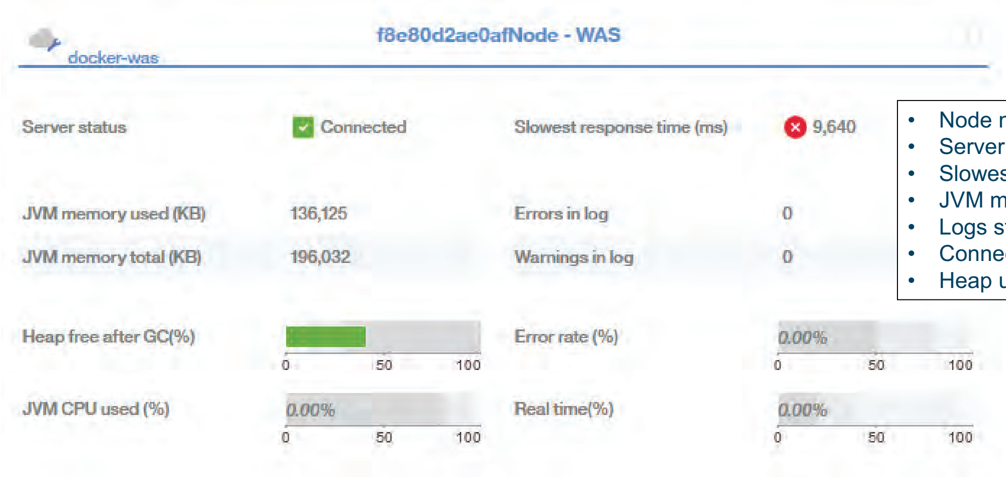
Monitors WebSphere Application Server, providing information on the following topics:

- Resource and performance information:
  - Server status
  - JVM usage
  - Thread Pool
  - Connect Pool
- Application performance:
  - Request response time
  - Web application information
- Java application server runtime environment
- Supports a range of environments, from single server to large deployments that require web tier clustering over multiple application server instances.

*Monitoring agent for WebSphere applications*

The monitoring agent for WebSphere applications returns data on resource usage and performance.

## WebSphere Application Server status overview



docker-was

f8e80d2ae0afNode - WAS

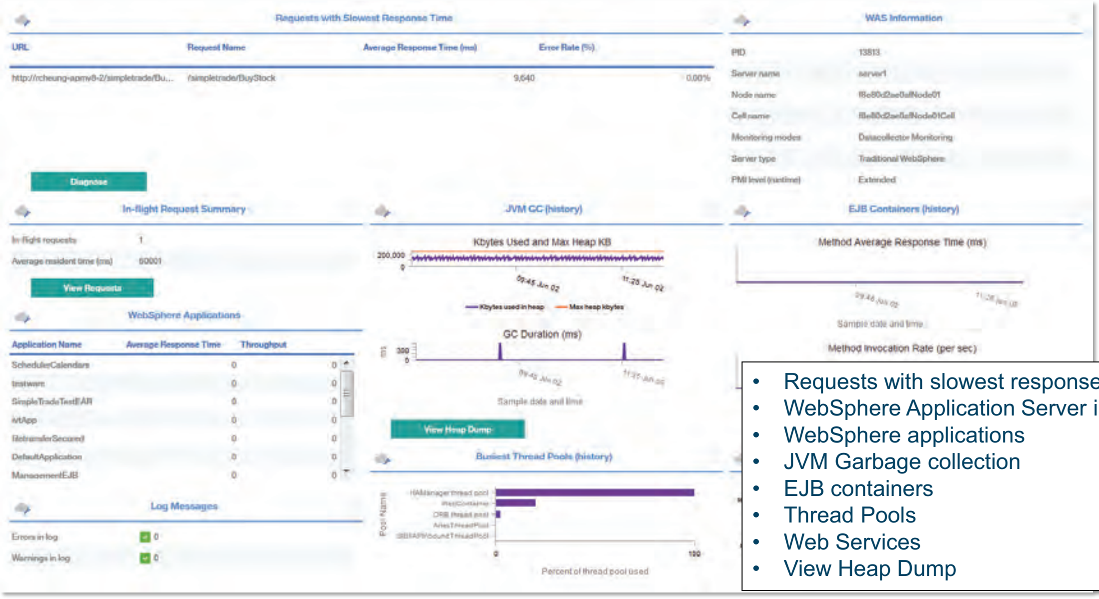
| | | | |
|---|---|---|---|
| Server status | ✓ Connected | Slowest response time (ms) | ✕ 9,640 |
| JVM memory used (KB) | 136,125 | Errors in log | 0 |
| JVM memory total (KB) | 196,032 | Warnings in log | 0 |
| Heap free after GC(%) | 🟩 | Error rate (%) | 0.00% |
| JVM CPU used (%) | 0.00% | Real time(%) | 0.00% |

- Node name (title bar)
- Server status
- Slowest response time (ms)
- JVM memory usage
- Logs status
- Connection pool status
- Heap usage

*WebSphere Application Server status overview*

The WebSphere Application Server status overview shows the health of the application server at a high level. You can see details about the Java virtual machine usage and information about database connector pool. Click anywhere on the status overview to see details of the server instance.

## WebSphere resource dashboard



Requests with Slowest Response Time

| URL | Request Name | Average Response Time (ms) | Error Rate (%) |
|---|---|---|---|
| http://rcheung-apmv8-2/simpletrade/Bu... | /simpletrade/BuyStock | 9,640 | 0.00% |

WAS Information

| | |
|---|---|
| PID | 13813 |
| Server name | server1 |
| Node name | f8e80d2ae0afNode01 |
| Cell name | f8e80d2ae0afNode01Cell |
| Monitoring modes | Datacollector Monitoring |
| Server type | Traditional WebSphere |
| PMI level (runtime) | Extended |

In-flight Request Summary

| | |
|---|---|
| In-flight requests | 1 |
| Average resident time (ms) | 60001 |

WebSphere Applications

| Application Name | Average Response Time | Throughput |
|---|---|---|
| SchedulerCalendars | 0 | 0 |
| testwars | 0 | 0 |
| SimpleTradeTestEAR | 0 | 0 |
| ivtApp | 0 | 0 |
| RetransferSecured | 0 | 0 |
| DefaultApplication | 0 | 0 |
| ManagementEJB | 0 | 0 |

Log Messages

| | |
|---|---|
| Errors in log | ✓ 0 |
| Warnings in log | ✓ 0 |

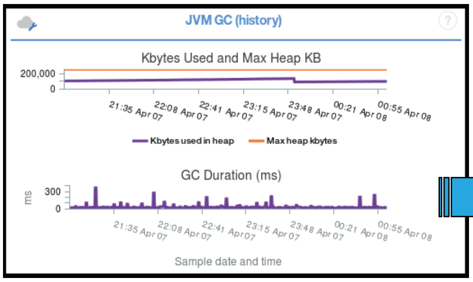
- Requests with slowest response time
- WebSphere Application Server information
- WebSphere applications
- JVM Garbage collection
- EJB containers
- Thread Pools
- Web Services
- View Heap Dump

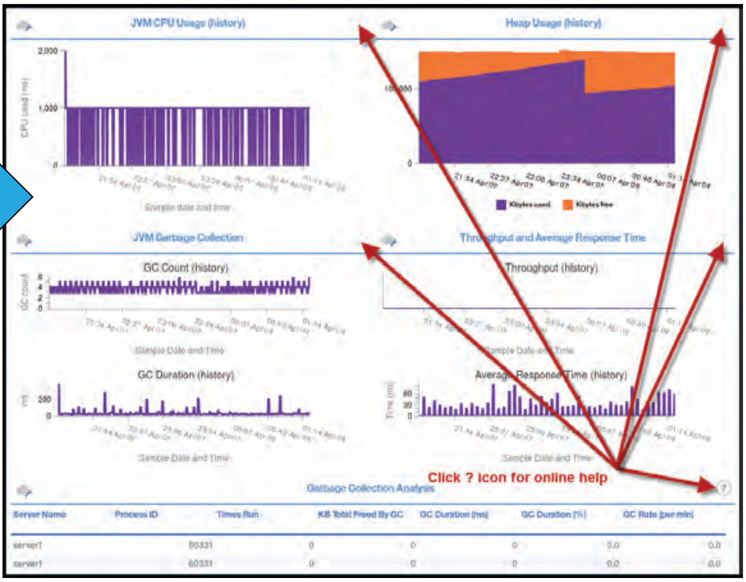
*WebSphere resource dashboard*

The main dashboard for the WebSphere agent offers resource usage information, as well as a summary of requests and their relative performance.

The WebSphere agent supports drill-down for all widgets except WAS Information. The Request Summary and In-flight Request Summary widgets are covered in Unit : .

## Garbage collection





| Server Name | Process ID | Times Run | KB Total Freed By GC | GC Duration (ms) | GC Duration (%) | GC Rate (per min) |
|---|---|---|---|---|---|---|
| server1 | 80331 | 0 | 0 | 0 | 0.0 | 0.0 |
| server1 | 60331 | 0 | 0 | 0 | 0.0 | 0.0 |

More detail dashboards available:

- WebSphere queries
- EJB containers
- Messaging and messaging engines
- Thread pools
- Database connection pools
- Web services

25

*Garbage collection*

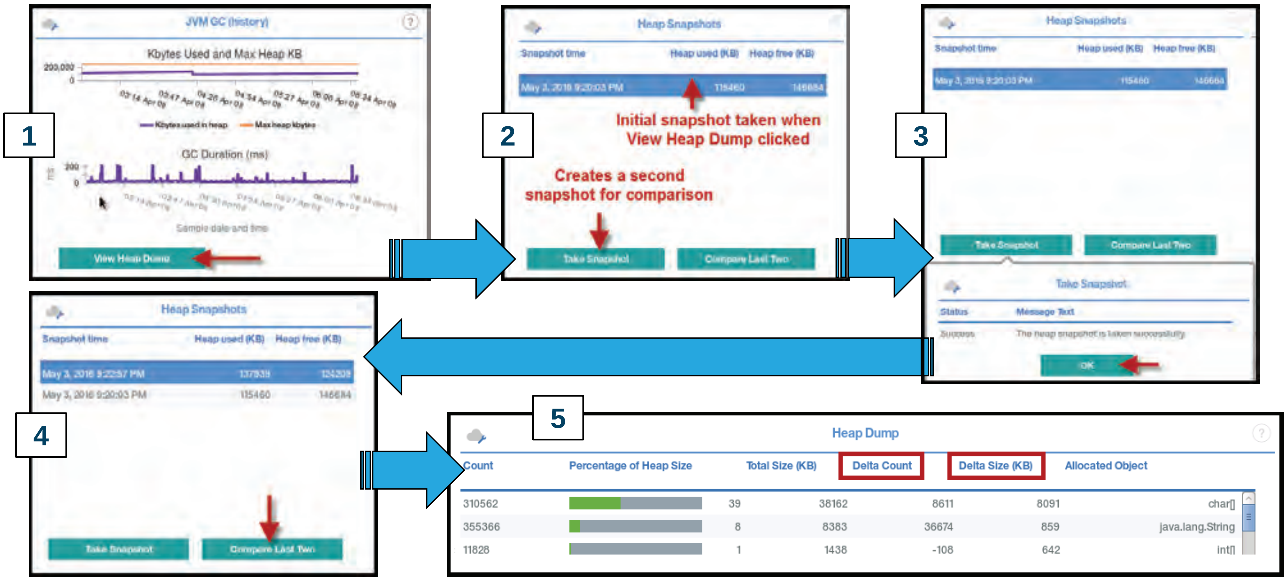
The JVM GC pane provides more details about garbage collection and heap usage. Some of the charts on this pane cover the past 24 hours, and some show the past 4 hours. Look carefully at the pane heading to understand the duration of the data being plotted.

You can put the mouse pointer over the plot chart points and hold the left mouse key down. Move the mouse to the right or left, and a small window shows the value and the date and time that it is recorded.

Garbage Collection is just one of the detail panes that you can invoke from the instance Overview pane. You can return directly to the Overview pane by clicking that hyperlink. Any time you see a pane that shows the hand icon when you hold your mouse pointer over it, that is, a link to another page.

Click the question mark (?) icon at the upper right of a widget to see online help for that set of performance attributes.

## Heap dump



Heap Snapshots

| Snapshot time | Heap used (KB) | Heap free (KB) |
|---|---|---|
| May 3, 2016 9:20:03 PM | 115460 | 146684 |

Initial snapshot taken when View Heap Dump clicked

Creates a second snapshot for comparison

Take Snapshot

| Status | Message Text |
|---|---|
| Success | The heap snapshot is taken successfully |

Heap Snapshots

| Snapshot time | Heap used (KB) | Heap free (KB) |
|---|---|---|
| May 3, 2016 9:22:57 PM | 137839 | 124209 |
| May 3, 2016 9:20:03 PM | 115460 | 146684 |

Heap Dump

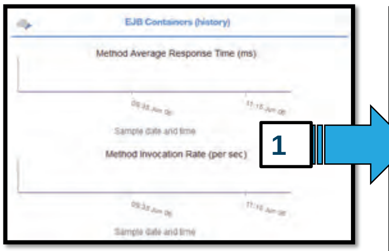
| Count | Percentage of Heap Size | Total Size (KB) | Delta Count | Delta Size (KB) | Allocated Object |
|---|---|---|---|---|---|
| 310562 | 39 | 38162 | 8611 | 8091 | char[] |
| 355366 | 8 | 8383 | 36674 | 859 | java.lang.String |
| 11828 | 1 | 1438 | -108 | 642 | int[] |

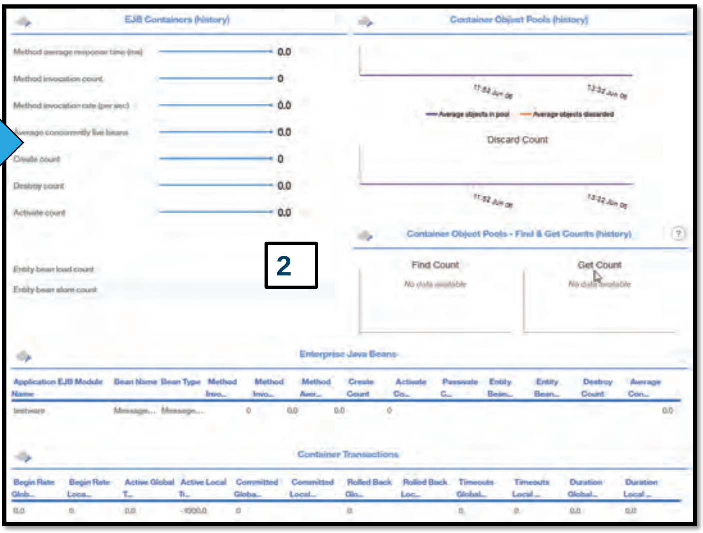
*Heap dump*

You can use the Heap Dump feature to perform these tasks:

- View what is currently on the heap (see #1 on the slide)
- Take a snapshot of the heap (see #2 and #3on the slide)
- Compare two heap snapshots (see #s 4 and 5,on the slide)
- View a single heap snapshot (not shown)

## EJB container





Method average response time (ms) 0.0
Method invocation count 0
Method invocation rate (per sec) 0.0
Average concurrently live beans 0.0
Create count 0
Destroy count 0.0
Activate count 0.0

Entity bean load count
Entity bean store count

Enterprise Java Beans

| Application Name | EJB Module | Bean Name | Bean Type | Method Invo... | Method Invo... | Method Aver... | Create Count | Activate Co... | Passivate C... | Entity Bean... | Entity Bean... | Destroy Count | Average Con... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| testware | | Message... | Message... | 0 | 0.0 | 0.0 | 0 | | | | | | 0.0 |

Container Transactions

| Begin Rate Glob... | Begin Rate Loca... | Active Global T... | Active Local Tr... | Committed Globa... | Committed Local... | Rolled Back Glo... | Rolled Back Loc... | Timeouts Global... | Timeouts Local ... | Duration Global... | Duration Local ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | 0 | 0.0 | -1000.0 | 0 | | 0 | | 0 | 0 | 0.0 | 0.0 |

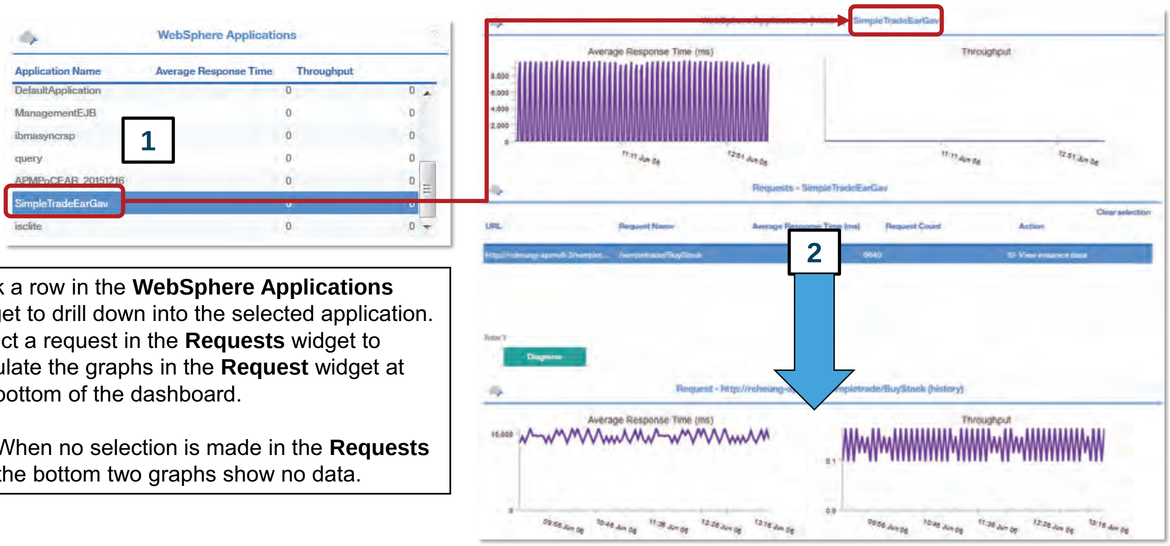
1. Click anywhere in the **EJB Containers** widget to access the details dashboard.
2. Review the available widgets.

**NOTE:** This screen capture was taken on a lightly loaded test system. The lack of data is normal.

*EJB container*

The EJB Containers details dashboard helps administrators diagnose the behavior of Enterprise Java Beans, or EJBs.

## WebSphere applications



WebSphere Applications

| Application Name | Average Response Time | Throughput |
|---|---|---|
| DefaultApplication | 0 | 0 |
| ManagementEJB | 0 | 0 |
| ibmasyncrsp | 0 | 0 |
| query | 0 | 0 |
| APMPoCEAR_20151216 | 0 | 0 |
| SimpleTradeEarGav | 0 | 0 |
| isclite | 0 | 0 |

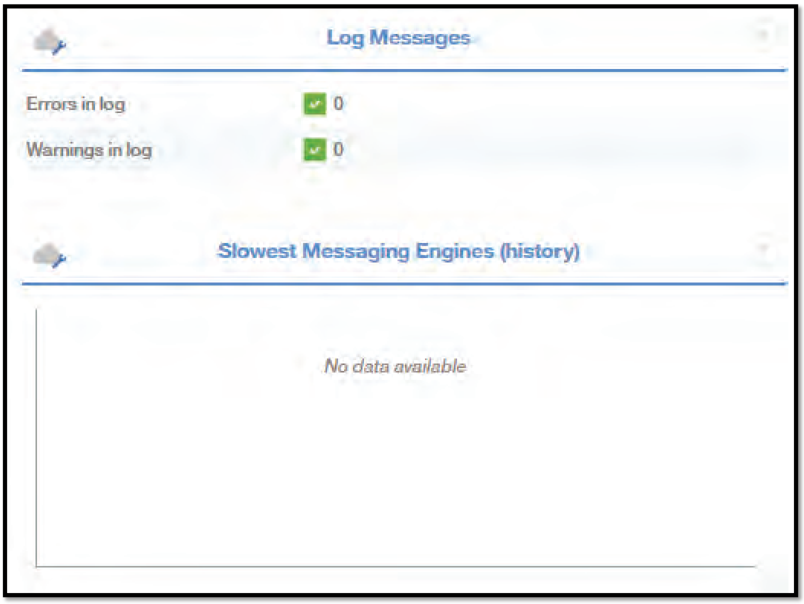
1. Click a row in the **WebSphere Applications** widget to drill down into the selected application.
2. Select a request in the **Requests** widget to populate the graphs in the **Request** widget at the bottom of the dashboard.

**NOTE:** When no selection is made in the **Requests** widget, the bottom two graphs show no data.

*WebSphere applications*

The WebSphere Applications widget affords access to request details. The Diagnose feature is covered in Unit : .

## Log messages and messaging engines



Log Messages

| | |
|---|---|
| Errors in log | ✓ 0 |
| Warnings in log | ✓ 0 |

Slowest Messaging Engines (history)

*No data available*

*Log messages and messaging engines*

The Log Messages widget displays the number of error and warning messages out of the most recent 100 log messages in the **SystemOut.log** file, or in the case of WebSphere Application Server Liberty, the messages.log file. When the error count is not zero, click this widget to access messages.

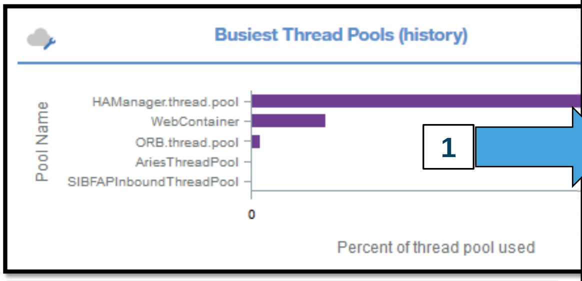
The Slowest Messaging Engines (history) widget displays the five message engines with the highest local wait time during the history period.

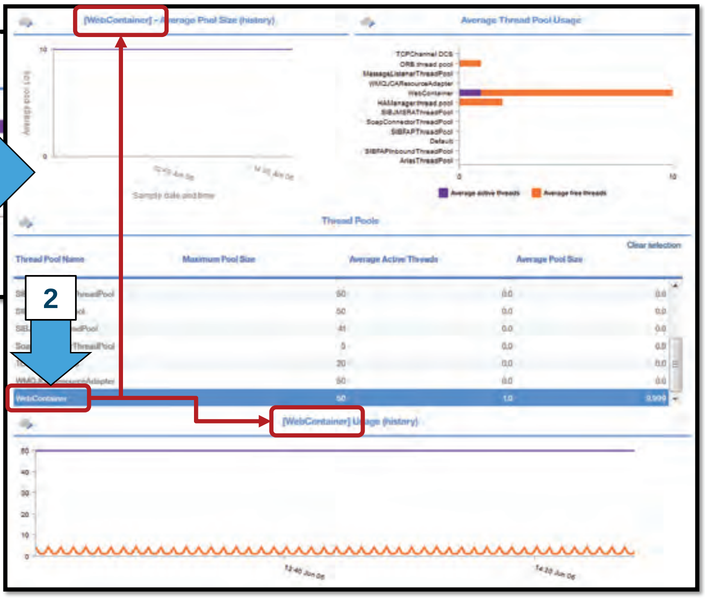
**Attention:** This group widget is not applicable to WebSphere Portal Server.

No data is available if one of these conditions is true:

- PMI is not enabled or is not set high enough on the application server. Some PMI data is only available after the application server is restarted.
- No data is available if the application does not use Messaging Engines.

## Busiest thread pools





1. Click anywhere in the **Busiest Thread Pools** widget to access the details dashboard.
2. Select a thread pool in the widget at the center of the dashboard,
3. Note the change in the header of the widgets at the bottom and upper left of the dashboard.

**NOTE:** The data that is displayed in the widget at the upper right remains constant regardless of which thread pool is selected.

30

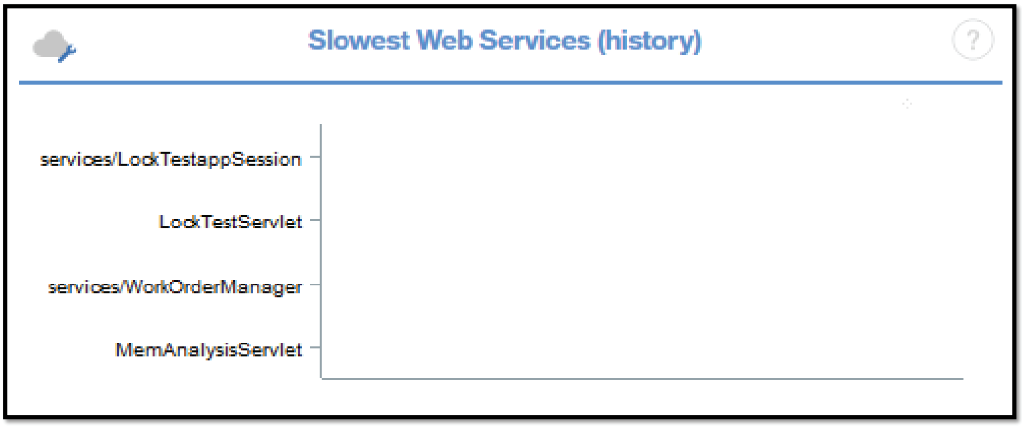
*Busiest thread pools*

The Busiest Thread Pools widget displays the five busiest thread pools during the history period.

No data is available if one of these conditions is true:

- PMI is not enabled or is not set high enough on the application server.
- Historical collection must be enabled for the Thread Pools attribute group to see data.
- Data is unavailable if you are monitoring WebSphere Application Server Liberty until the default executor thread pool is used by a request.

## Slowest web services


Slowest Web Services (history)
?
services/LockTestappSession
LockTestServlet
services/WorkOrderManager
MemAnalysisServlet

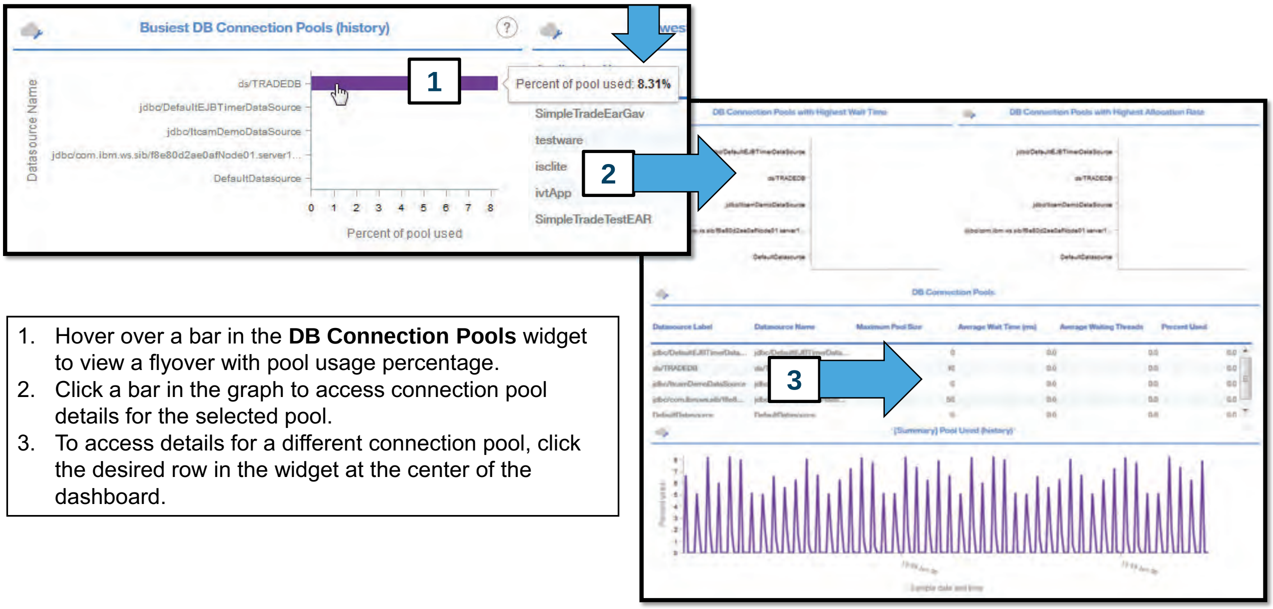
*Slowest web services*

The Slowest Web Services widget displays the five web services with the slowest average response time during the history period.

No data is available if one of these conditions is true:

- PMI is not enabled or is not set high enough on the application server. Some PMI data is only available after the application server is restarted.
- No data is displayed if an application does not use web services.

## Busiest DB Connection pools



1. Hover over a bar in the **DB Connection Pools** widget to view a flyover with pool usage percentage.
2. Click a bar in the graph to access connection pool details for the selected pool.
3. To access details for a different connection pool, click the desired row in the widget at the center of the dashboard.

*Busiest DB Connection pools*

These key performance indicators are included on DB Connection Pools dashboard:

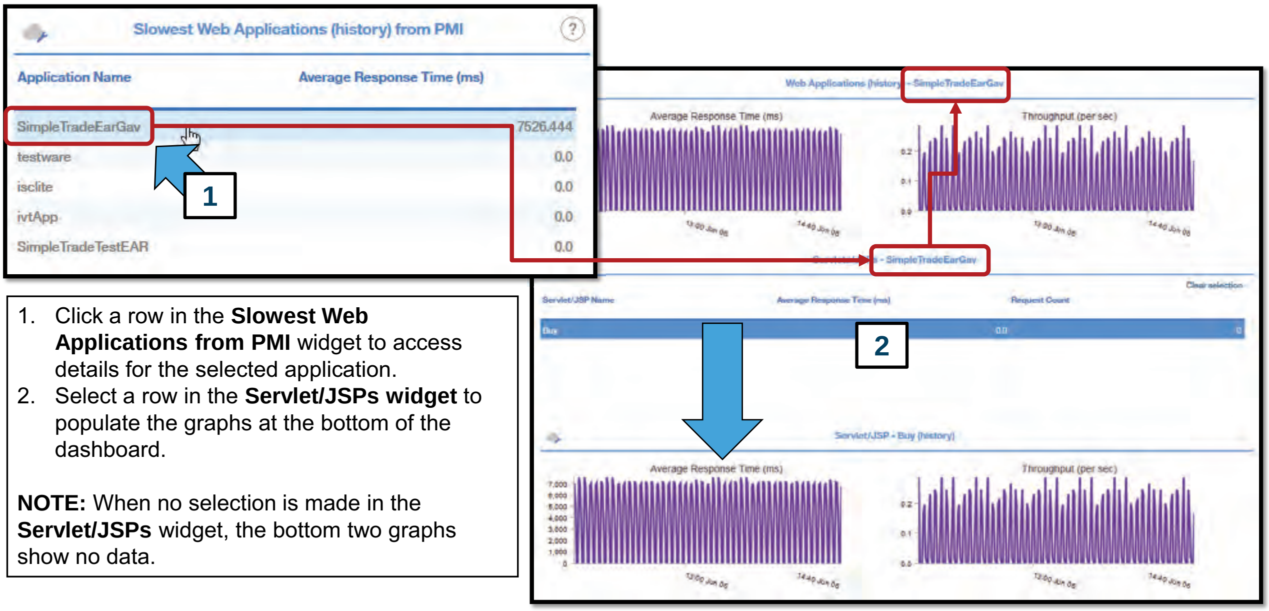
- Average Wait Time (ms)

  The average time (in milliseconds) a client waited for a connection; blank if no transactions are completed during the interval.

- Average Waiting Threads

  The average number of threads waiting for a connection during the interval. This value is an average of several values collected over an interval.

## Slowest web applications from PMI



Slowest Web Applications (history) from PMI

| Application Name | Average Response Time (ms) |
|---|---|
| SimpleTradeEarGav | 7526.444 |
| testware | 0.0 |
isclite | 0.0 |
| ivtApp | 0.0 |
| SimpleTradeTestEAR | 0.0 |

1
2
Web Applications (history) - SimpleTradeEarGav
Average Response Time (ms)
Throughput (per sec)
Servlet/JSPs - SimpleTradeEarGav
Clear selection
Servlet/JSP Name
Average Response Time (ms)
Request Count
Buy
0.0
0
Servlet/JSP - Buy (history)
Average Response Time (ms)
Throughput (per sec)

1. Click a row in the **Slowest Web Applications from PMI** widget to access details for the selected application.
2. Select a row in the **Servlet/JSPs widget** to populate the graphs at the bottom of the dashboard.

**NOTE:** When no selection is made in the **Servlet/JSPs** widget, the bottom two graphs show no data.

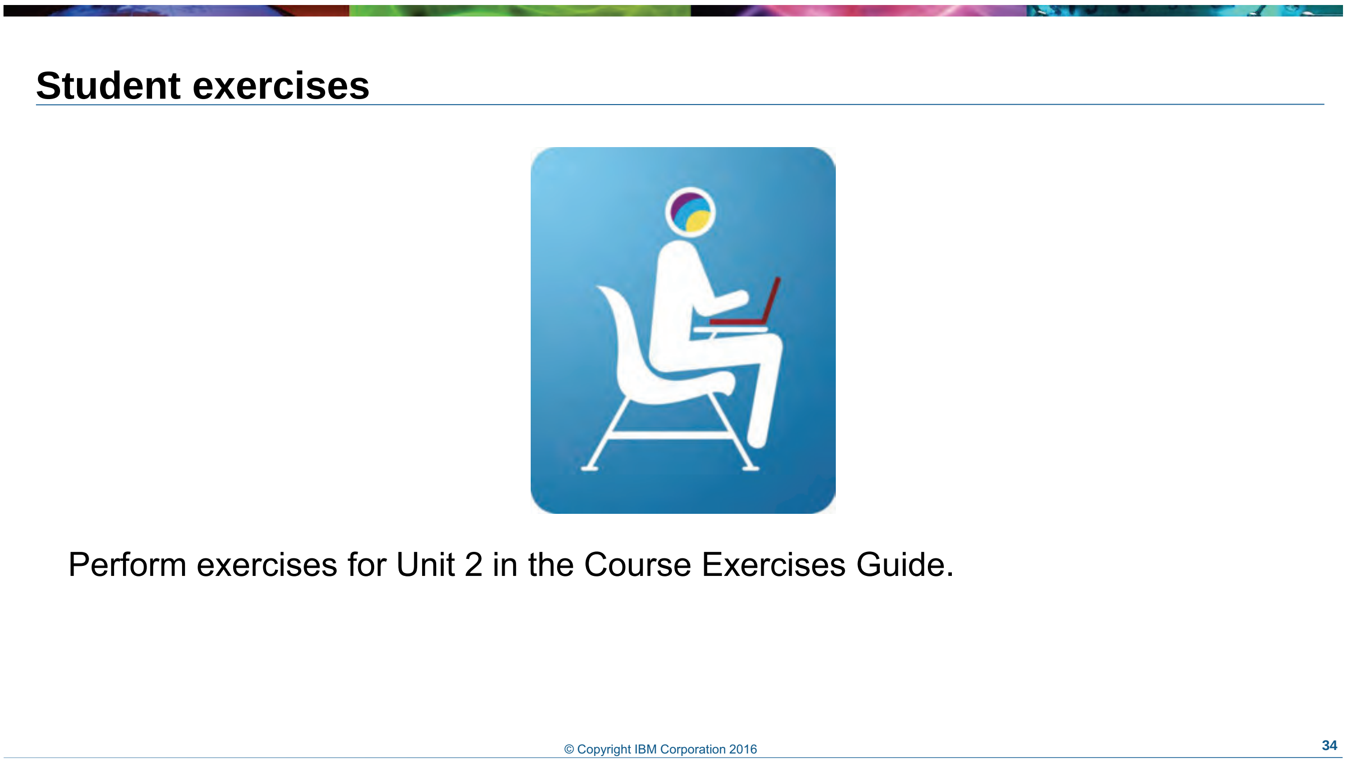
*Slowest web applications from PMI*

The Slowest Web Applications (history) from PMI widget displays the five web applications with the slowest average response time during the history period.

For WebSphere Portal Servers, the group widget displays data for all web applications that were discovered during the current interval. The data is limited to the slowest 100 web applications.

No data is available if one of these conditions is true:

- PMI is not enabled or is not set high enough on the application server.
- Historical collection must be enabled for the Web Applications attribute group to see data.

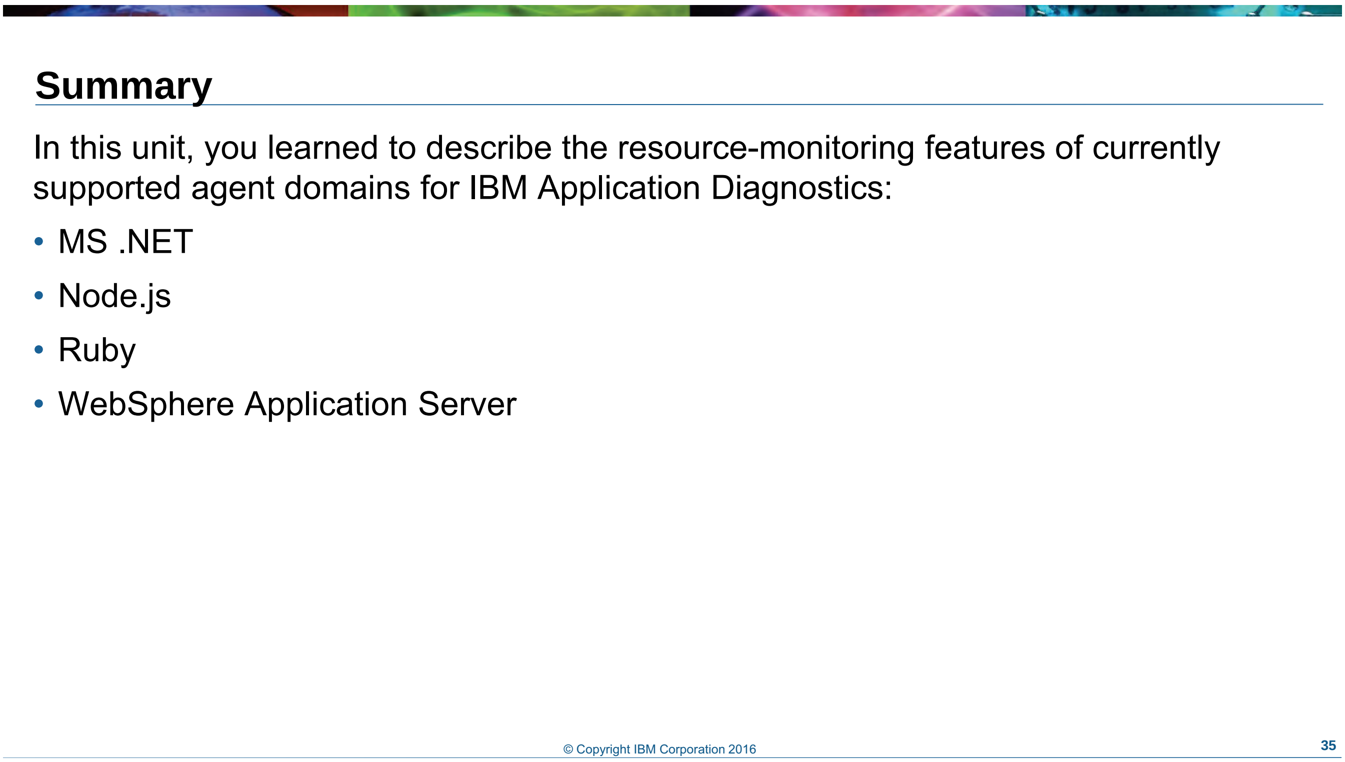


## Student exercises

Perform exercises for Unit 2 in the Course Exercises Guide.

*Student exercises*

You now perform the exercises for Unit 2 in the Course Exercises Guide.



## Summary

In this unit, you learned to describe the resource-monitoring features of currently supported agent domains for IBM Application Diagnostics:

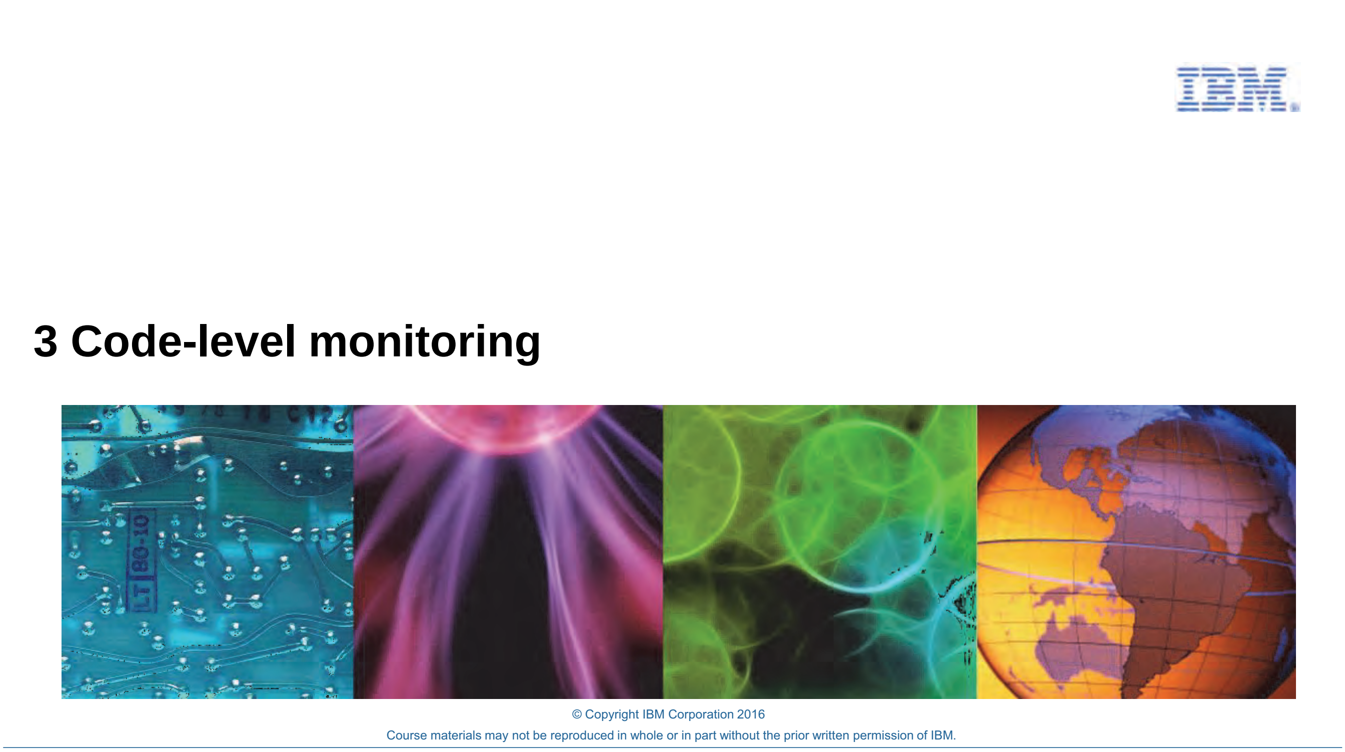
- MS .NET
- Node.js
- Ruby
- WebSphere Application Server

*Summary*

In this unit, you learned to describe the resource-monitoring features of currently supported agent domains for IBM Application Diagnostics:

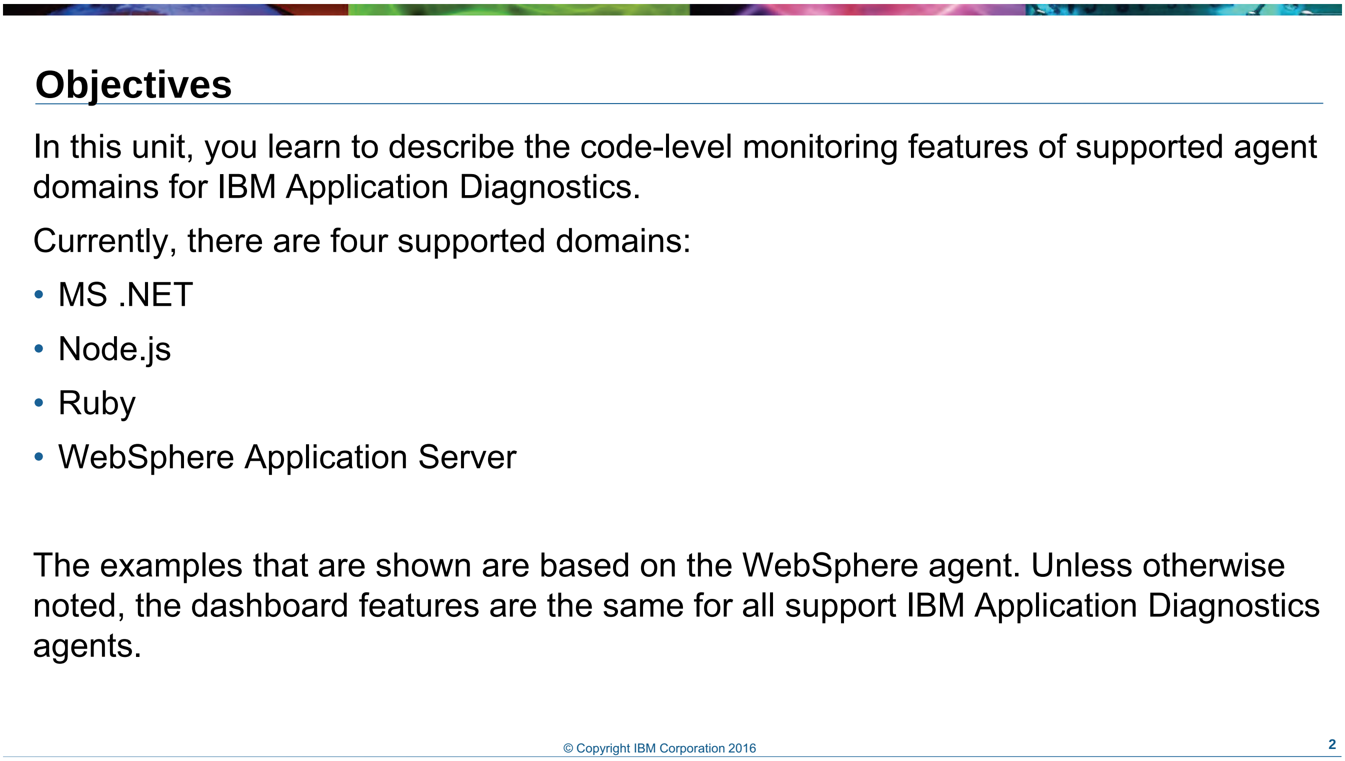
- MS .NET
- Node.js
- Ruby
- WebSphere Application Server

# 3 Code-level monitoring



## 3 Code-level monitoring

This presentation is a technical overview of the code-level monitoring features of IBM Application Diagnostics.



## Objectives

In this unit, you learn to describe the code-level monitoring features of supported agent domains for IBM Application Diagnostics.

Currently, there are four supported domains:

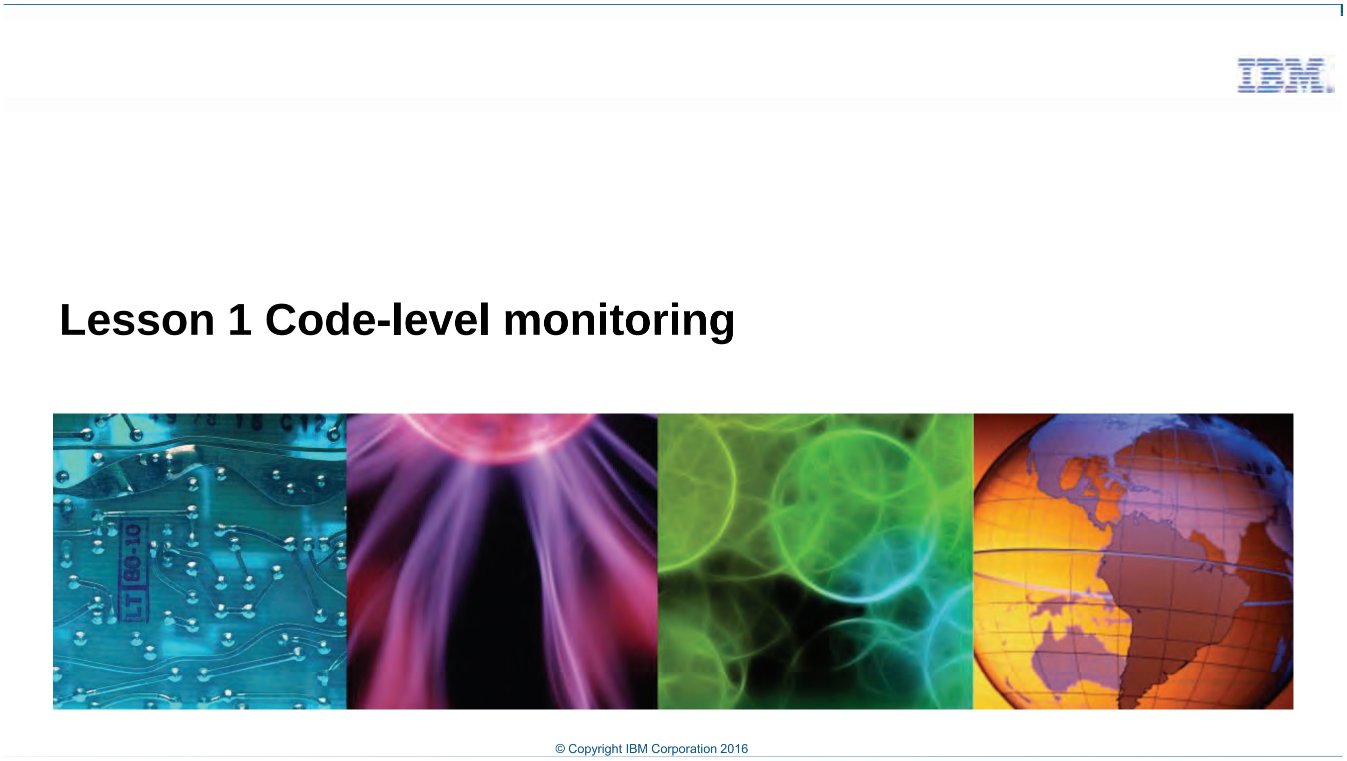
- MS .NET
- Node.js
- Ruby
- WebSphere Application Server

The examples that are shown are based on the WebSphere agent. Unless otherwise noted, the dashboard features are the same for all support IBM Application Diagnostics agents.

*Objectives*

In this unit, you learn to describe the code-level monitoring features of supported agent domains for IBM Application Diagnostics.

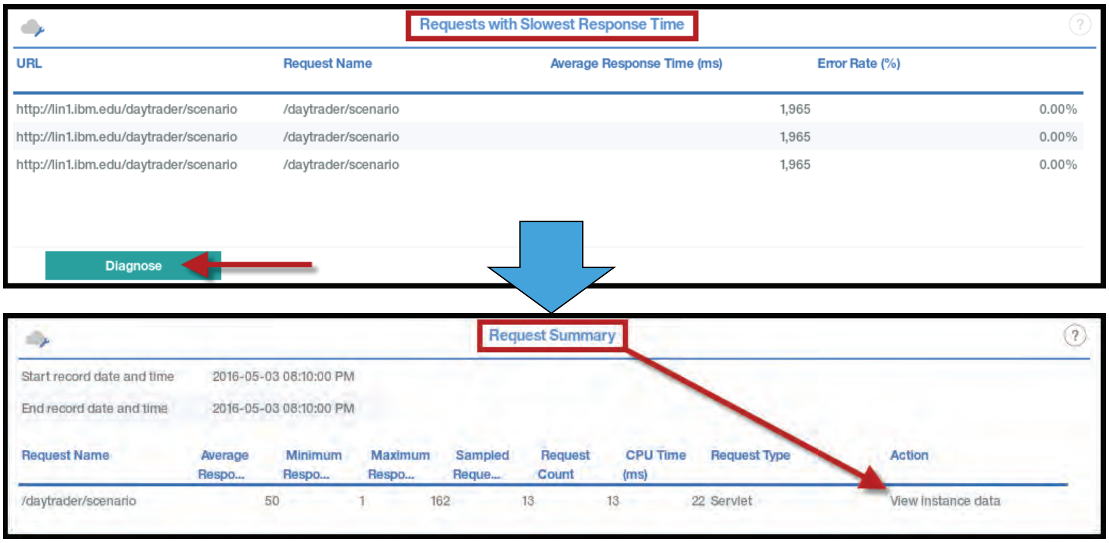
# Lesson 1  Code-level monitoring



Lesson 1 Code-level monitoring

In this section of this presentation, you get a description of code-level monitoring features common to all IBM Application Diagnostics agents.

## Viewing code-level data



Requests with Slowest Response Time

| URL | Request Name | Average Response Time (ms) | Error Rate (%) |
|---|---|---|---|
| http://lin1.ibm.edu/daytrader/scenario | /daytrader/scenario | 1,965 | 0.00% |
| http://lin1.ibm.edu/daytrader/scenario | /daytrader/scenario | 1,965 | 0.00% |
| http://lin1.ibm.edu/daytrader/scenario | /daytrader/scenario | 1,965 | 0.00% |

Request Summary

Start record date and time 2016-05-03 08:10:00 PM

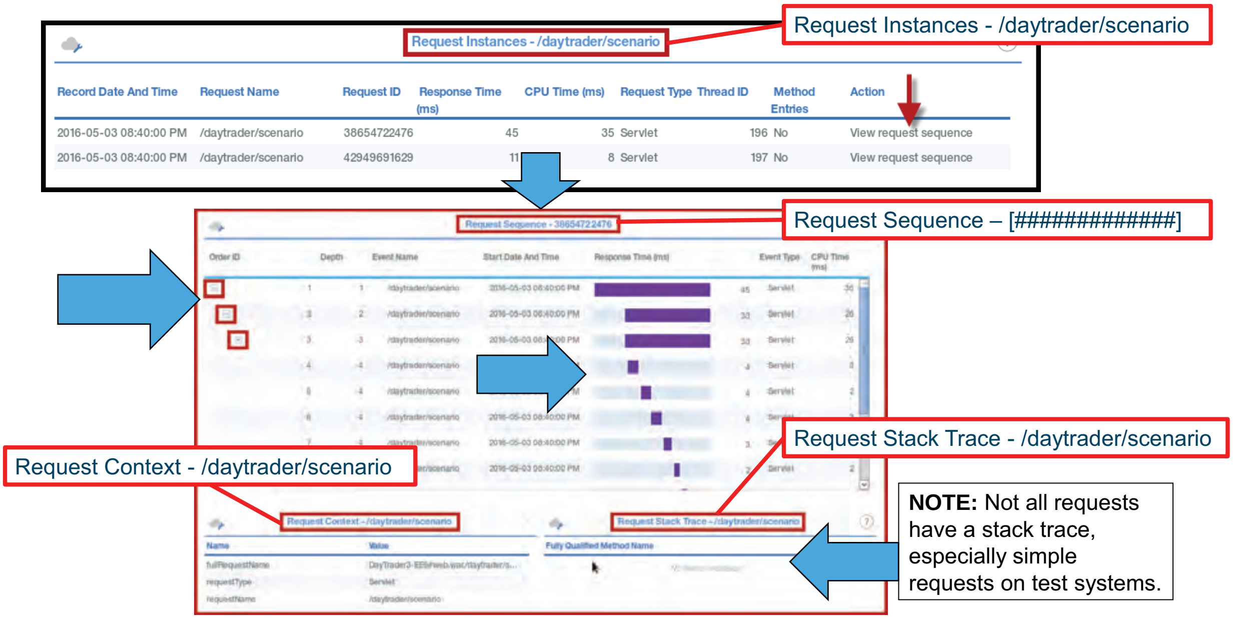
End record date and time 2016-05-03 08:10:00 PM

| Request Name | Average Respo... | Minimum Respo... | Maximum Respo... | Sampled Reque... | Request Count | CPU Time (ms) | Request Type | Action |
|---|---|---|---|---|---|---|---|---|
| /daytrader/scenario | 50 | 1 | 162 | 13 | 13 | 22 | Servlet | View instance data |

*Viewing code-level data*

Click **Diagnose** as the first step in code-level diagnosis. On the next screen, click **View instance data** in the Action column for the request that you want to examine. You continue drilling down on the next slide.

# Viewing code-level data (Cont.)



Request Instances - /daytrader/scenario

| Record Date And Time | Request Name | Request ID | Response Time (ms) | CPU Time (ms) | Request Type | Thread ID | Method Entries | Action |
|---|---|---|---|---|---|---|---|---|
| 2016-05-03 08:40:00 PM | /daytrader/scenario | 38654722476 | 45 | 35 | Servlet | 196 | No | View request sequence |
| 2016-05-03 08:40:00 PM | /daytrader/scenario | 42949691629 | 11 | 8 | Servlet | 197 | No | View request sequence |

Request Sequence – [############]

Request Sequence - 38654722476

| Order ID | Depth | Event Name | Start Date And Time | Response Time (ms) | Event Type | CPU Time (ms) |
|---|---|---|---|---|---|---|
| 1 | 1 | /daytrader/scenario | 2016-05-03 08:40:00 PM | 45 | Servlet | 36 |
| 2 | 2 | /daytrader/scenario | 2016-05-03 08:40:00 PM | 33 | Servlet | 26 |
| 3 | 3 | /daytrader/scenario | 2016-05-03 08:40:00 PM | 33 | Servlet | 26 |
| 4 | 4 | /daytrader/scenario | [illegible] | 4 | Servlet | 0 |
| 5 | 4 | /daytrader/scenario | [illegible] | 4 | Servlet | 2 |
| 6 | 4 | /daytrader/scenario | 2016-05-03 08:40:00 PM | 4 | [illegible] | [illegible] |
| 7 | 4 | /daytrader/scenario | 2016-05-03 08:40:00 PM | 3 | [illegible] | [illegible] |
| [illegible] | [illegible] | [illegible]er/scenario | 2016-05-03 08:40:00 PM | 2 | Servlet | 2 |

Request Context - /daytrader/scenario

Request Context - /daytrader/scenario

| Name | Value |
|---|---|
| fullRequestName | DayTrader3-EE6#web.war/daytrader/s... |
| requestType | Servlet |
| requestName | /daytrader/scenario |

Request Stack Trace - /daytrader/scenario

Request Stack Trace - /daytrader/scenario

Fully Qualified Method Name

**NOTE:** Not all requests have a stack trace, especially simple requests on test systems.

© Copyright IBM Corporation 2016

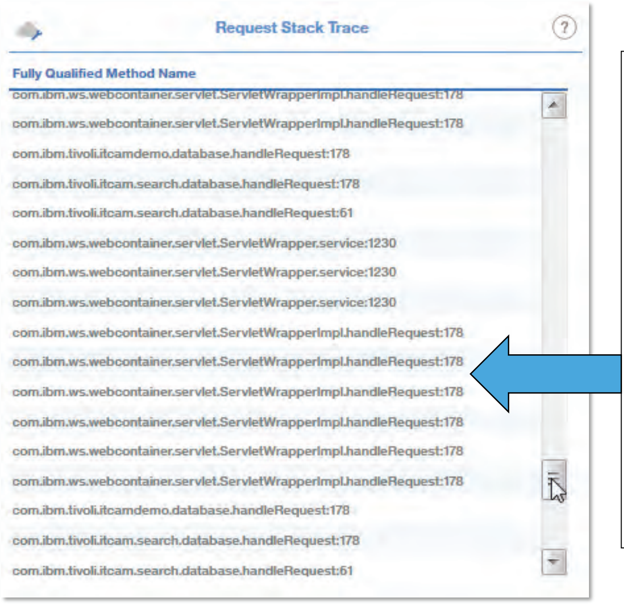
5

*Viewing code-level data (Cont.)*

On the Request Instances widget, locate the instance that you want to examine, and click **View request sequence** in the Action column.

On the next screen, the request sequence shows the order of events that make up the request. Click the plus signs to expand the request. Click individual requests in the sequence to view associated request context and request stack trace.

## Understanding the request stack trace



```
Fully Qualified Method Name
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.tivoli.itcamdemo.database.handleRequest:178
com.ibm.tivoli.itcam.search.database.handleRequest:178
com.ibm.tivoli.itcam.search.database.handleRequest:61
com.ibm.ws.webcontainer.servlet.ServletWrapper.service:1230
com.ibm.ws.webcontainer.servlet.ServletWrapper.service:1230
com.ibm.ws.webcontainer.servlet.ServletWrapper.service:1230
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178
com.ibm.tivoli.itcamdemo.database.handleRequest:178
com.ibm.tivoli.itcam.search.database.handleRequest:178
com.ibm.tivoli.itcam.search.database.handleRequest:61
```

The **Request Stack Trace** widget displays the stack trace for the selected request or nested request. The data collector tracks the progress of a request. When a violation occurs, the data collector captures a stack trace.

Use the stack trace to identify and fix looping conditions.

The stack trace also provides the depth of a method call. A high number of nested calls might lead to a high method response time.

The number at the end of each line is the line of application source code.

**NOTE:** Lines numbers are not available for calls to system libraries. Instead of an integer, the string “Unknown” is appended to the method for system calls.

*Understanding the request stack trace*

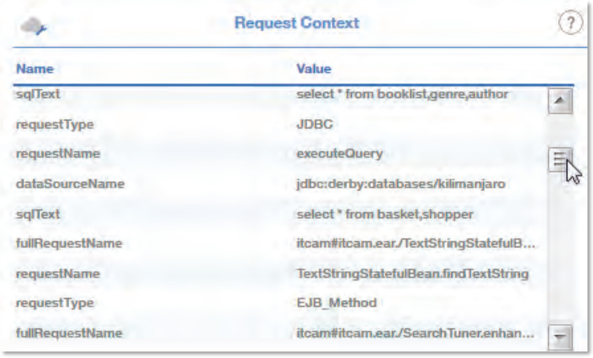
Stack trace data is unavailable if any of these conditions is true:

- The data collector within the JVM of the application server is not configured.
- Diagnostics mode is disabled.
- The collection of stack trace data for the request type is disabled.
- The collection threshold value was not exceeded, even if the collection of stack trace data is enabled.

The Performance Management console supports various key performance indicators (KPIs).

The fully qualified method name for each Java object that handled the request instance and the line number that was invoked in each class. The data type is *String*.

## Understanding the request context



Request Context

| Name | Value |
|---|---|
| sqlText | select * from booklist,genre,author |
| requestType | JDBC |
| requestName | executeQuery |
| dataSourceName | jdbc:derby:databases/kilimanjaro |
| sqlText | select * from basket,shopper |
| fullRequestName | itcam#itcam.ear./TextStringStatefulB... |
| requestName | TextStringStatefulBean.findTextString |
| requestType | EJB_Method |
| fullRequestName | itcam#itcam.ear./SearchTuner.enhan... |

The **Request Context** widget displays the Java context of each request (event name).

The type of information in the request context depends on the request type (event type):

- For a JNDI request, the request context includes the JNDI resource URI.
- For a JDBC request, the request context includes the data source name, the request name, and an SQL statement.
- For an EJB method and servlet requests, the request context includes the request name and the application module name.

**NOTE:** When you have multiple instances of a resource in your environment, the request context helps you identify the resource used by the request instance.

*Understanding the request context*

Request context data is not available if any of these conditions is true:

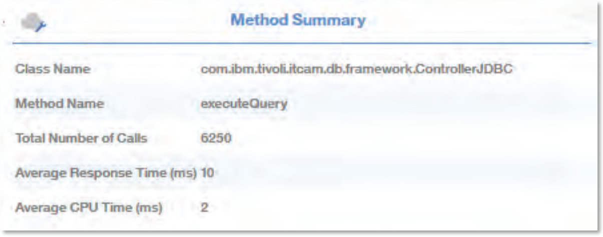
- Diagnostics mode is disabled.
- The collection of request context data is disabled.
- Threshold-based monitoring is enabled but a threshold is not exceeded.

# Lesson 2  Additional features of the WebSphere agent

**Lesson 2 Additional features of the WebSphere agent**

In this section, you get a description of additional code-level monitoring features that the WebSphere Application Server agent supplies.

## Understanding the method summary



Method Summary

| | |
|---|---|
| Class Name | com.ibm.tivoli.itcam.db.framework.ControllerJDBC |
| Method Name | executeQuery |
| Total Number of Calls | 6250 |
| Average Response Time (ms) | 10 |
| Average CPU Time (ms) | 2 |

The **Method Summary** widget displays a summary of the performance of the method type that you selected in the Request Sequence table.

Select an event in the Request Sequence table to display a summary of the performance of the method type.

**Supported KPIs:**
- Class Name
- Method Name
- Total Number of Calls
- Average Response Time (ms)
- Average CPU Time (ms)

*Understanding the method summary*

Method Summary data is unavailable if any of these conditions is true:

- Diagnostics mode is disabled.
- Method trace is disabled.
- The method is not of the Event type.

## *KPI descriptions*

### Class Name

The method class. Derived from the Class Name attribute of the Method Summary Specific data set. The data type is *String*.

### Method Name

The name of the method. Derived from the Class Name attribute of the Method Summary Specific data set (Method Summary Specific.Method Name). The data type is *String*.

### Total Number of Calls

The total number of times the method was called. Derived from the Total Method Calls attribute of the Method Summary Specific data set. The data type is *Integer* (32-bit numeric property) with enumerated values.

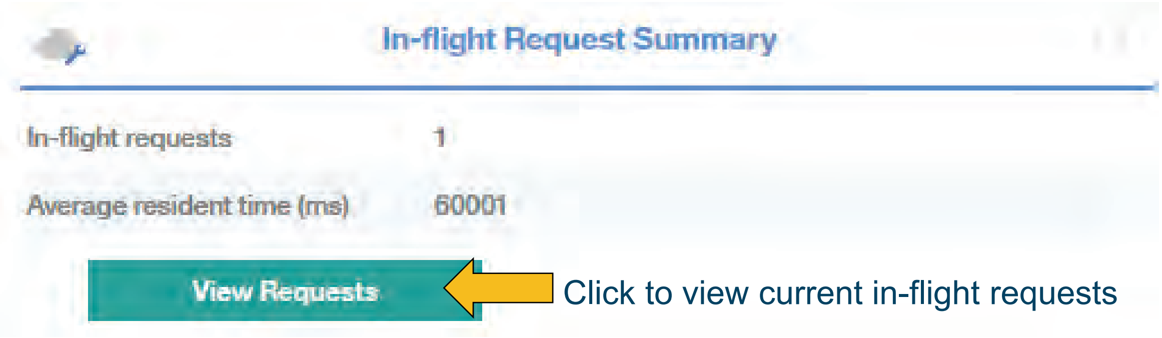
### Average Response Time (ms)

The average response time for all instances of this method. Derived from the Average Method Resp time attribute of the Method Summary Specific data set (Method Summary Specific.Average Method Response Time). The data type is *Integer* (32-bit numeric property) with enumerated values.

### Average CPU Time (ms)

The average CPU time for all instances of this method. Derived from the Average Method Cpu time attribute of the Method Summary Specific data set. The data type is *Integer* (32-bit numeric property) with enumerated values.

## In-flight requests



In-flight Request Summary

| | |
|---|---|
| In-flight requests | 1 |
| Average resident time (ms) | 60001 |

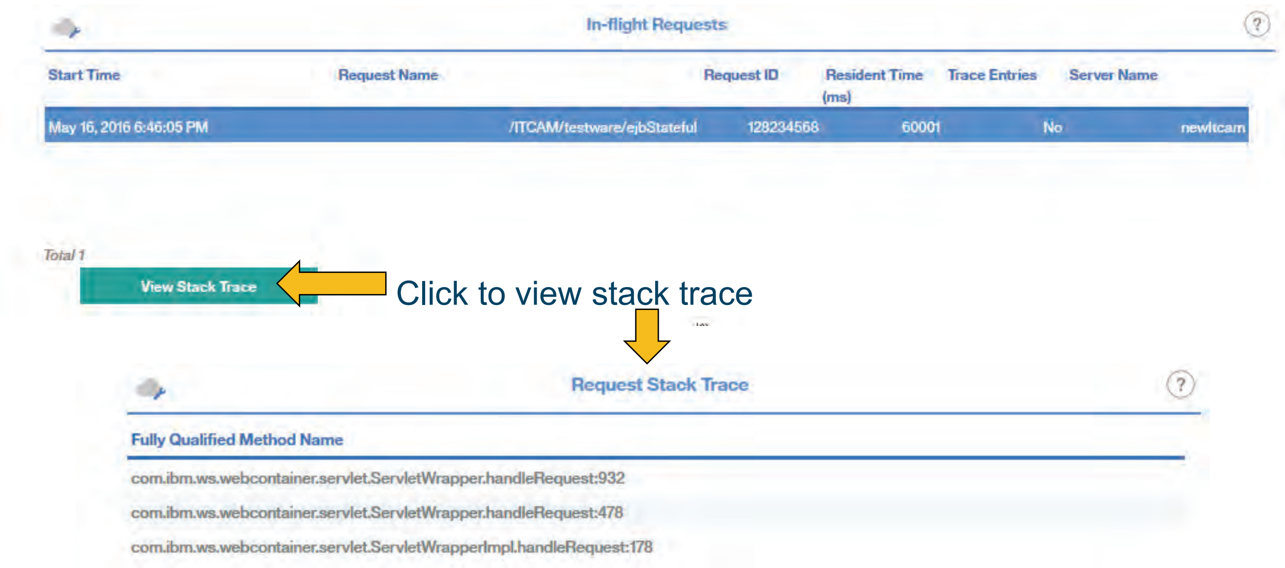
Click to view current in-flight requests

NOTE: By default, an in-flight request must take at least 60 seconds to complete before it shows in the widget.

*In-flight requests*

The minimum threshold to qualify as an in-flight request can be adjusted downward. However, a lower threshold can adversely affect performance.

## In-flight requests (Cont.)



In-flight Requests

| Start Time | Request Name | Request ID | Resident Time (ms) | Trace Entries | Server Name |
|---|---|---|---|---|---|
| May 16, 2016 6:46:05 PM | /ITCAM/testware/ejbStateful | 128234568 | 60001 | No | newitcam |

*Total 1*

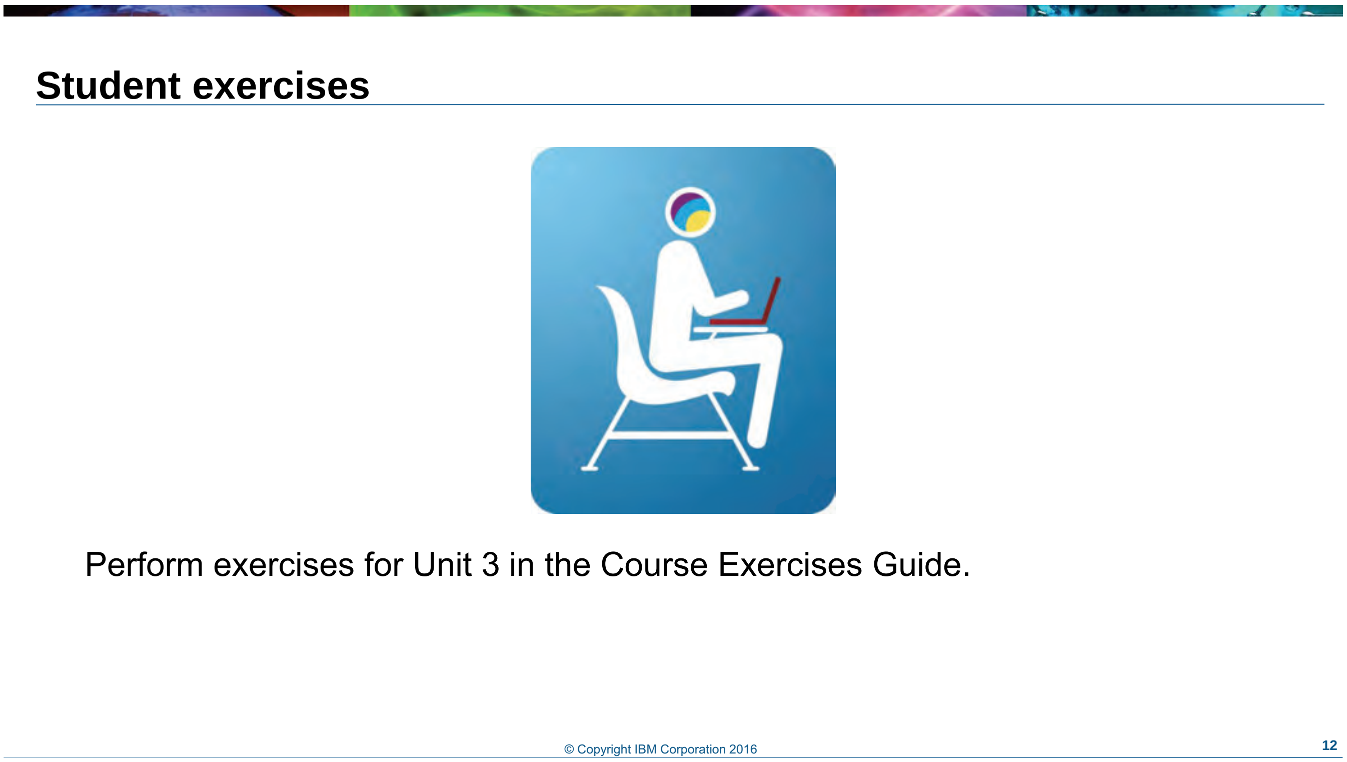
Click to view stack trace

Request Stack Trace

| Fully Qualified Method Name |
|---|
| com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest:932 |
| com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest:478 |
| com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest:178 |

*In-flight requests (Cont.)*

Select the in-flight request that you want to diagnose further; then click **View Stack Trace**.

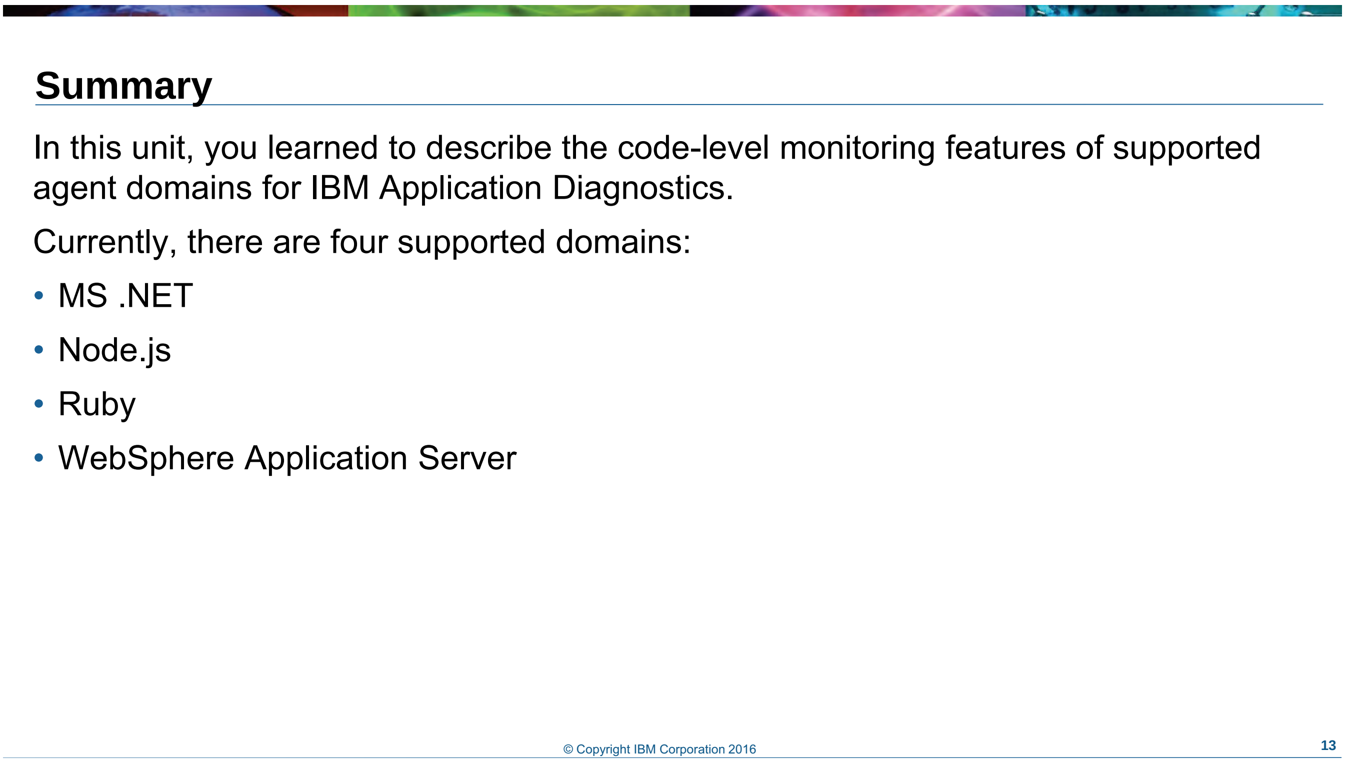


## Student exercises

Perform exercises for Unit 3 in the Course Exercises Guide.

*Student exercises*

You now perform exercises for Unit 3 in the Course Exercises Guide.



## Summary

In this unit, you learned to describe the code-level monitoring features of supported agent domains for IBM Application Diagnostics.

Currently, there are four supported domains:

- MS .NET
- Node.js
- Ruby
- WebSphere Application Server

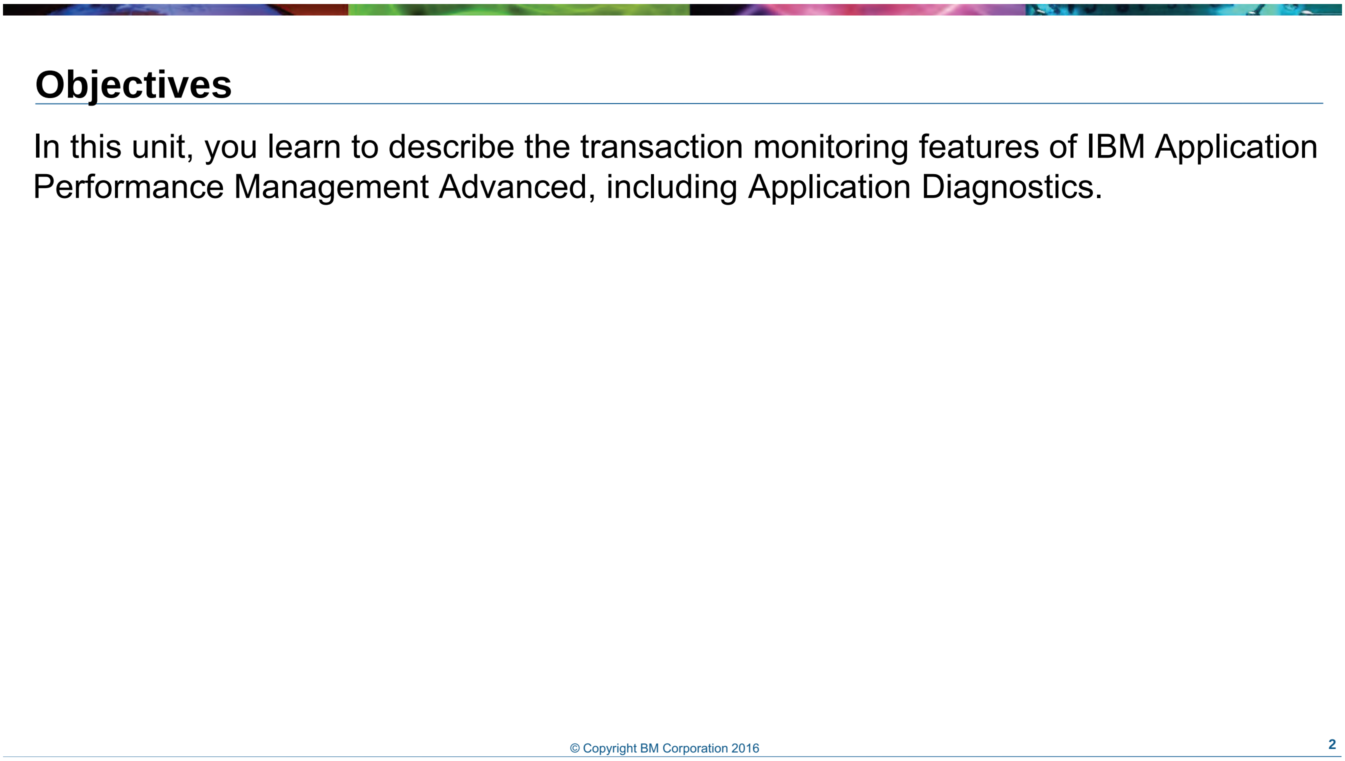
*Summary*

In this unit, you learned to describe the code-level monitoring features of supported agent domains for IBM Application Diagnostics.

# 4 Transaction tracking

## 4 Transaction tracking

This presentation is an overview of the transaction monitoring features of IBM Application Performance Management Advanced.



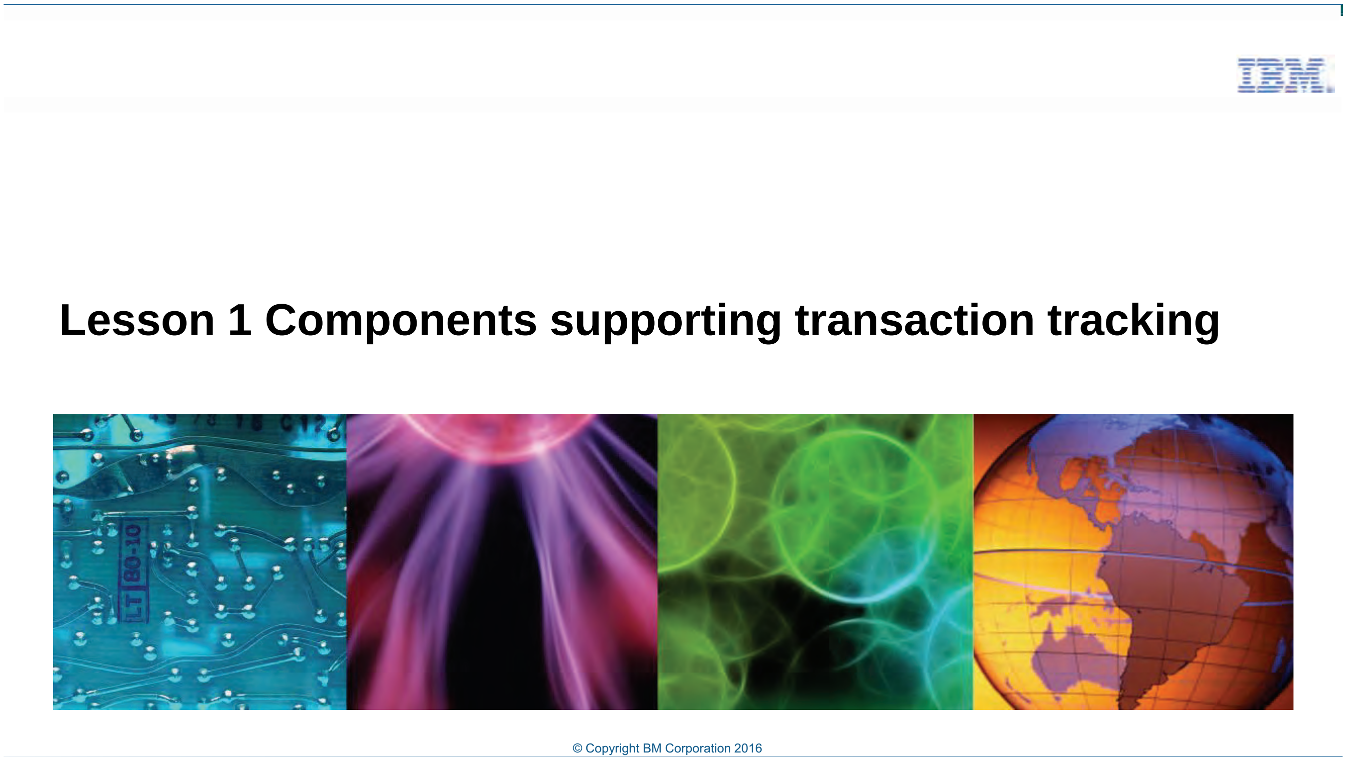
## Objectives

In this unit, you learn to describe the transaction monitoring features of IBM Application Performance Management Advanced, including Application Diagnostics.

*Objectives*

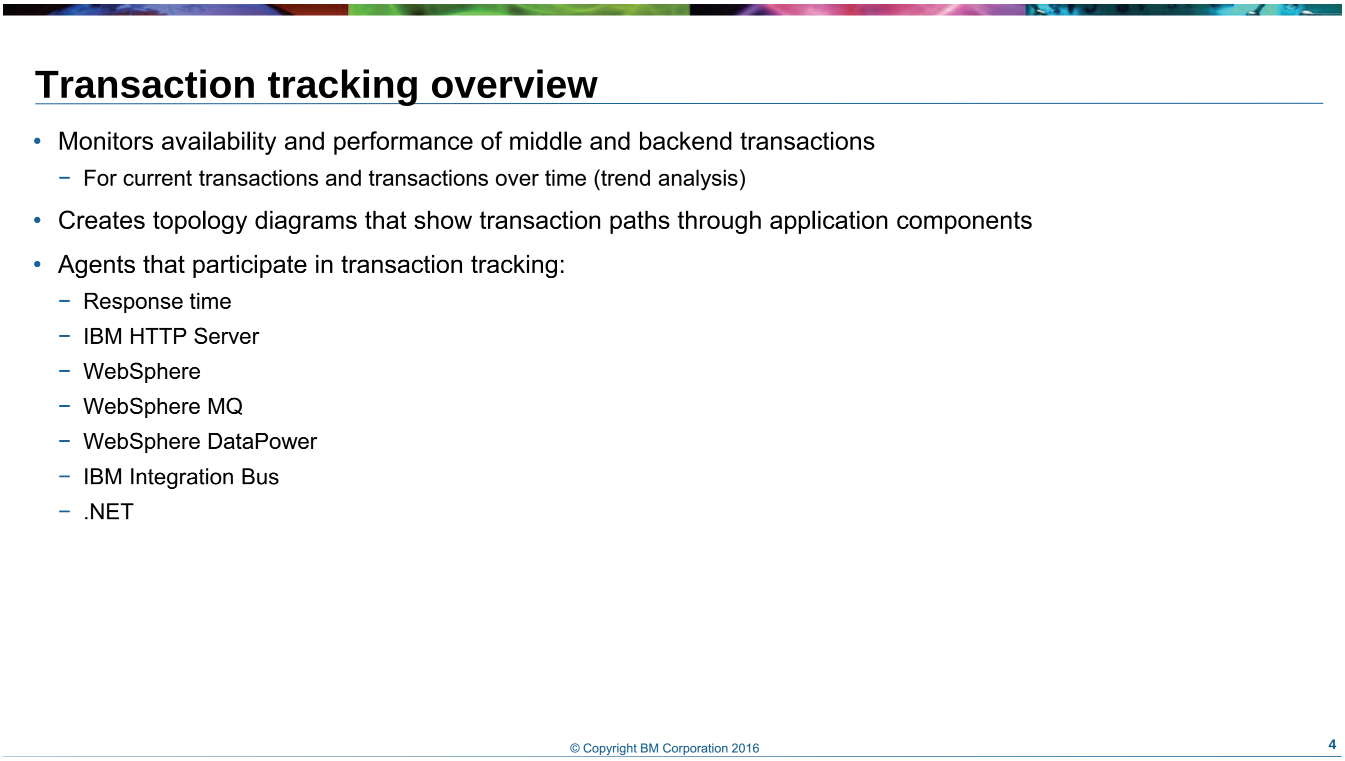
In this unit, you learn to describe the transaction monitoring features of IBM Application Performance Management Advanced, including Application Diagnostics.

# Lesson 1  Components supporting transaction tracking



## Lesson 1 Components supporting transaction tracking

This lesson provides an overview of the components involved in transaction tracking.



## Transaction tracking overview

- Monitors availability and performance of middle and backend transactions
  - For current transactions and transactions over time (trend analysis)
- Creates topology diagrams that show transaction paths through application components
- Agents that participate in transaction tracking:
  - Response time
  - IBM HTTP Server
  - WebSphere
  - WebSphere MQ
  - WebSphere DataPower
  - IBM Integration Bus
  - .NET

4

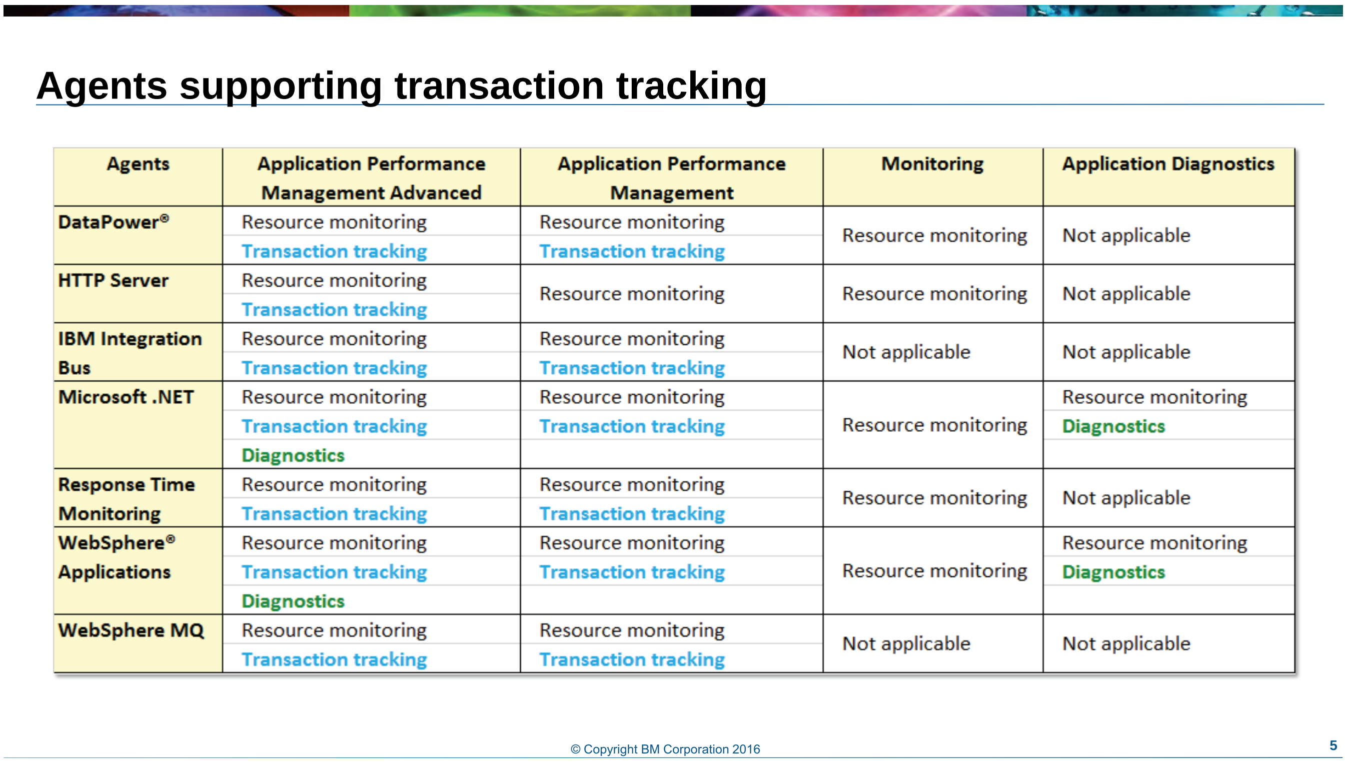
*Transaction tracking overview*

A transaction is a request for a service from an application:

- User request to the application (front end transaction)
- Application to application (middle or back-end transaction)

Transaction monitoring helps you determine answers to these questions:

- Which application has slow or failing transactions?
- Which transactions are slow or failing?
- Where in the transaction path is a problem occurring?
- What is the availability or performance of your application over time?
- What is the transaction load, current or over time, for your application?



## Agents supporting transaction tracking

| Agents | Application Performance Management Advanced | Application Performance Management | Monitoring | Application Diagnostics |
|---|---|---|---|---|
| DataPower® | Resource monitoring<br>Transaction tracking | Resource monitoring<br>Transaction tracking | Resource monitoring | Not applicable |
| HTTP Server | Resource monitoring<br>Transaction tracking | Resource monitoring | Resource monitoring | Not applicable |
| IBM Integration Bus | Resource monitoring<br>Transaction tracking | Resource monitoring<br>Transaction tracking | Not applicable | Not applicable |
| Microsoft .NET | Resource monitoring<br>Transaction tracking<br>Diagnostics | Resource monitoring<br>Transaction tracking | Resource monitoring | Resource monitoring<br>Diagnostics |
| Response Time Monitoring | Resource monitoring<br>Transaction tracking | Resource monitoring<br>Transaction tracking | Resource monitoring | Not applicable |
| WebSphere® Applications | Resource monitoring<br>Transaction tracking<br>Diagnostics | Resource monitoring<br>Transaction tracking | Resource monitoring | Resource monitoring<br>Diagnostics |
| WebSphere MQ | Resource monitoring<br>Transaction tracking | Resource monitoring<br>Transaction tracking | Not applicable | Not applicable |

5

*Agents supporting transaction tracking*

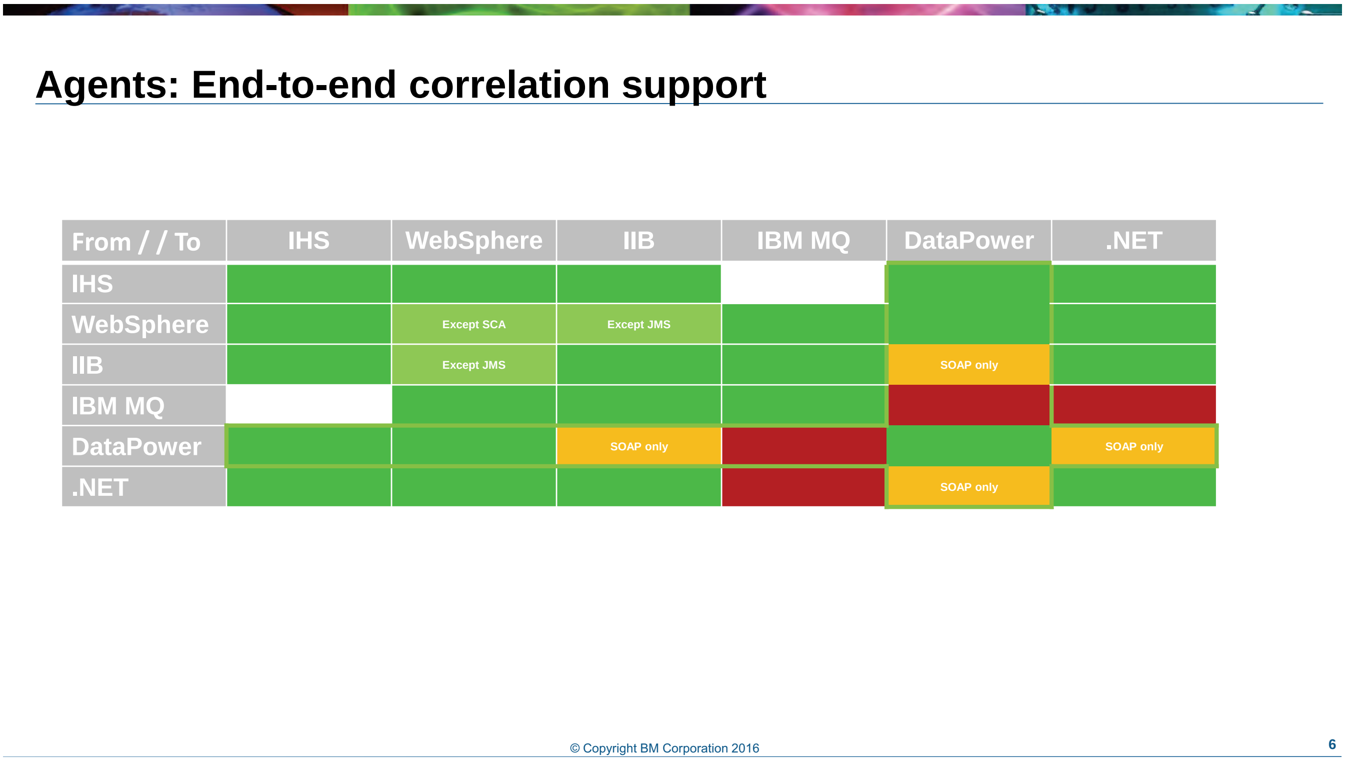
This slide shows which agents support transaction tracking and resource monitoring.

Two of those agents also support code-level diagnostics. Code-level diagnostic data collection is available only in IBM Application Diagnostics, which is included in IBM Application Performance Management Advanced.

Two additional agents that are part of Application Diagnostics support code-level diagnostics:

- Node.js monitoring agent
- Ruby monitoring agent

At present, these agents do not support transaction tracking.



## Agents: End-to-end correlation support

| From / / To | IHS | WebSphere | IIB | IBM MQ | DataPower | .NET |
|---|---|---|---|---|---|---|
| IHS | 🟩 | 🟩 | 🟩 | ⬜ | 🟩 | 🟩 |
| WebSphere | 🟩 | 🟩 Except SCA | 🟩 Except JMS | 🟩 | 🟩 | 🟩 |
| IIB | 🟩 | 🟩 Except JMS | 🟩 | 🟩 | 🟨 SOAP only | 🟩 |
| IBM MQ | ⬜ | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 |
| DataPower | 🟩 | 🟩 | 🟨 SOAP only | 🟥 | 🟩 | 🟨 SOAP only |
| .NET | 🟩 | 🟩 | 🟩 | 🟥 | 🟨 SOAP only | 🟩 |

© Copyright BM Corporation 2016

6

*Agents: End-to-end correlation support*

This slide shows whether transactions can be tracked between the different agents that can do transaction tracking. Green indicates yes. Red indicates no. Yellow indicates yes in limited cases and lists the cases. White indicates that this intersection is not applicable.

Service Component Architecture (SCA) is a specification that describes a model for building applications and systems using a service-oriented architecture (SOA). SCA simplifies the creation and integration of business applications that are built using an SOA by separating business logic from its implementation so that you can focus on assembling an integrated application without knowing details of its implementation.

SCA divides the steps of building a service-oriented application into two major parts:

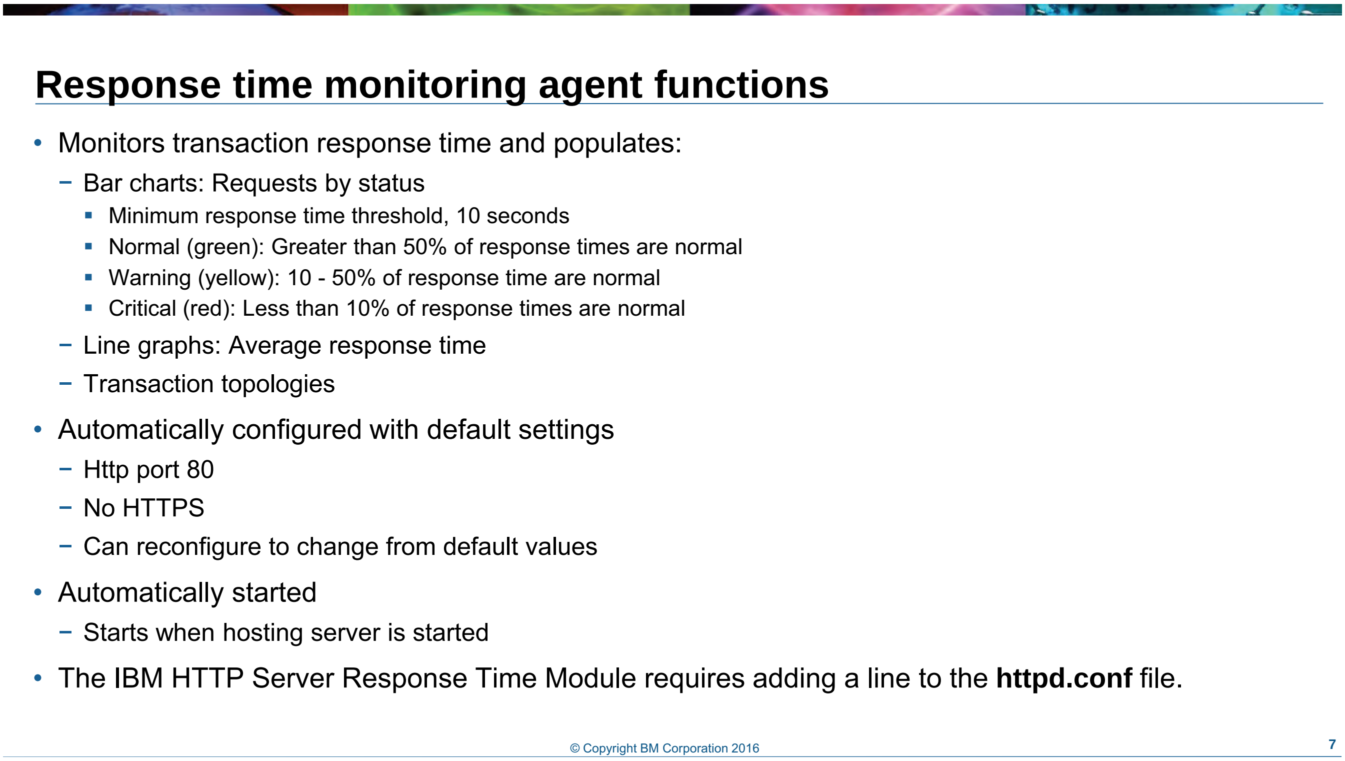
- The implementation of components that provide services and use other services.
- The assembly of these service components to build the business application.

You can assemble service components graphically with the WebSphere Integration Developer and add the implementation later.

The JMS protocol is based on the Java Message Service (JMS) and transfers messages through transactional, persistent JMS queues provided by, for example, IBM WebSphere MQ. The JMS protocol supports the following JMS message types:

- StreamMessage (as a byte array)
- BytesMessage (as a byte array)
- TextMessage

In the JMS protocol, one system sends a JMS message to another. After the second system receives the message, it removes it from the queue. From this point forward, the receiving system can process the message asynchronously.



## Response time monitoring agent functions

- Monitors transaction response time and populates:
  - Bar charts: Requests by status
    - Minimum response time threshold, 10 seconds
    - Normal (green): Greater than 50% of response times are normal
    - Warning (yellow): 10 - 50% of response time are normal
    - Critical (red): Less than 10% of response times are normal
  - Line graphs: Average response time
  - Transaction topologies
- Automatically configured with default settings
  - Http port 80
  - No HTTPS
  - Can reconfigure to change from default values
- Automatically started
  - Starts when hosting server is started
- The IBM HTTP Server Response Time Module requires adding a line to the **httpd.conf** file.

© Copyright BM Corporation 2016

7

*Response time monitoring agent functions*

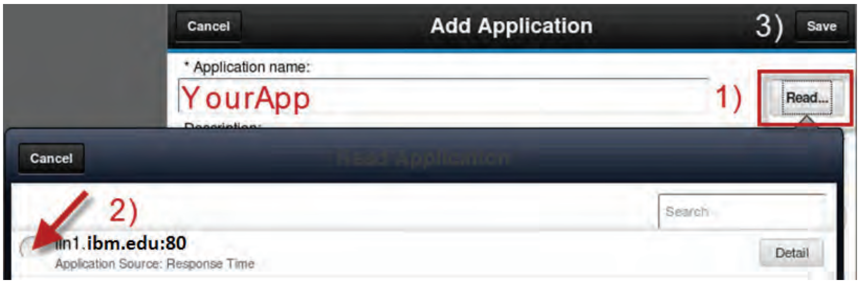
The response time monitoring agent works in combination with the monitored resource agent to produce application topologies. Currently, only the .NET and WebSphere Application Server agents support topologies.

When all features are enabled, the top-level dashboard contains a bar chart that shows aggregated transactions over the past 2 hours.In addition, an aggregated transaction topology shows request elements.

The response time agent is a single instance per monitored system. It is configured with default settings and starts automatically.

## Incorporating the Response Time Agent in applications

The Response Time agent must be incorporated in applications with the **Read** feature in the Application Editor.



* Application name: YourApp

lin1.ibm.edu:80
Application Source: Response Time

With your application open in the Application Editor, perform these steps:

1. Click **Read**.
2. Select the Application Source: Response Time item.
3. Click **Save**.

*Incorporating the Response Time Agent in applications*

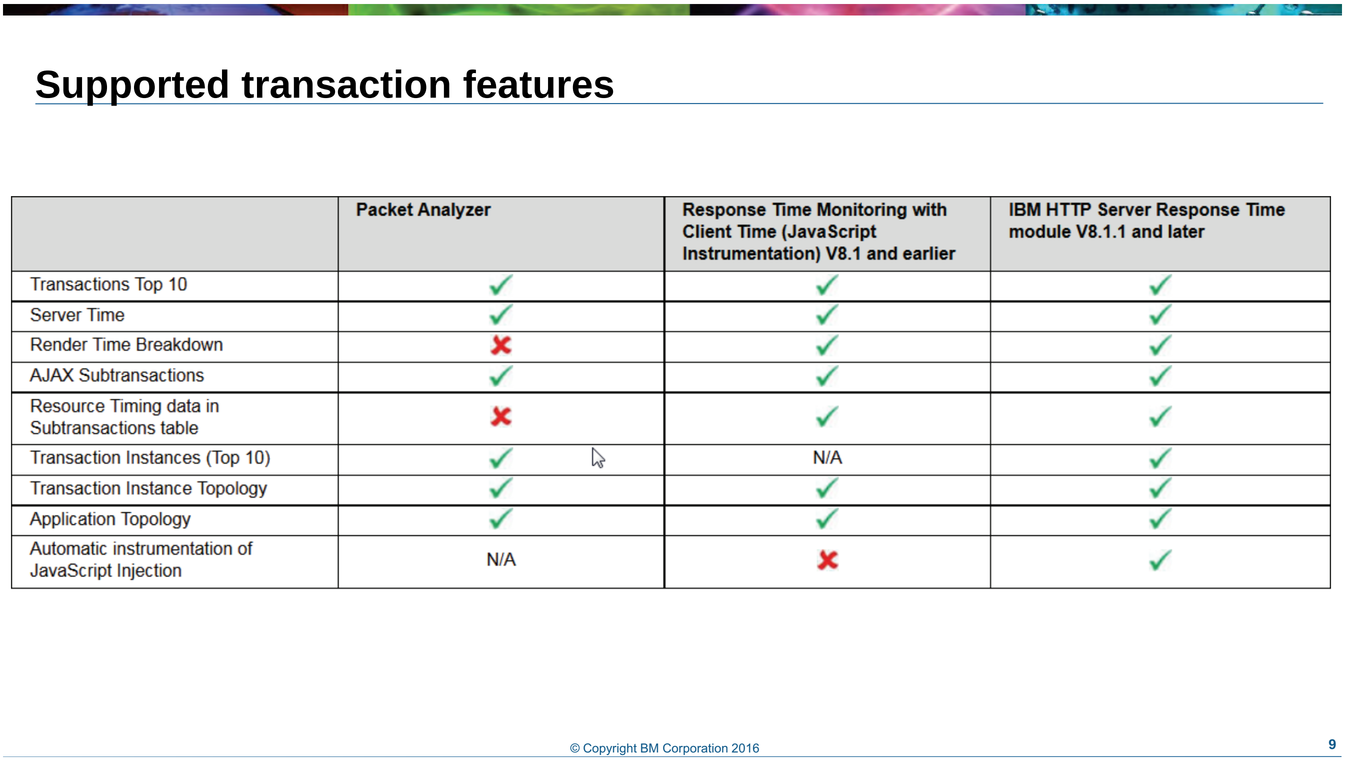
The Response Time agent must be incorporated in applications with the Read feature in the Application Editor.

With your application open in the Application Editor, perform these steps:

1. 1Click **Read**.
2. Select the Application Source: Response Time item.

**Note:** Make sure the fully qualified version is present. In this course, the application server is lin1.ibm.edu. If the only available Response Time instance is **lin1:80** (80 is the port number), generate more data until you see the fully qualified version, **lin1.ibm.edu:80**.

3. Select the fully qualified host name.
4. Click **Save**.



## Supported transaction features

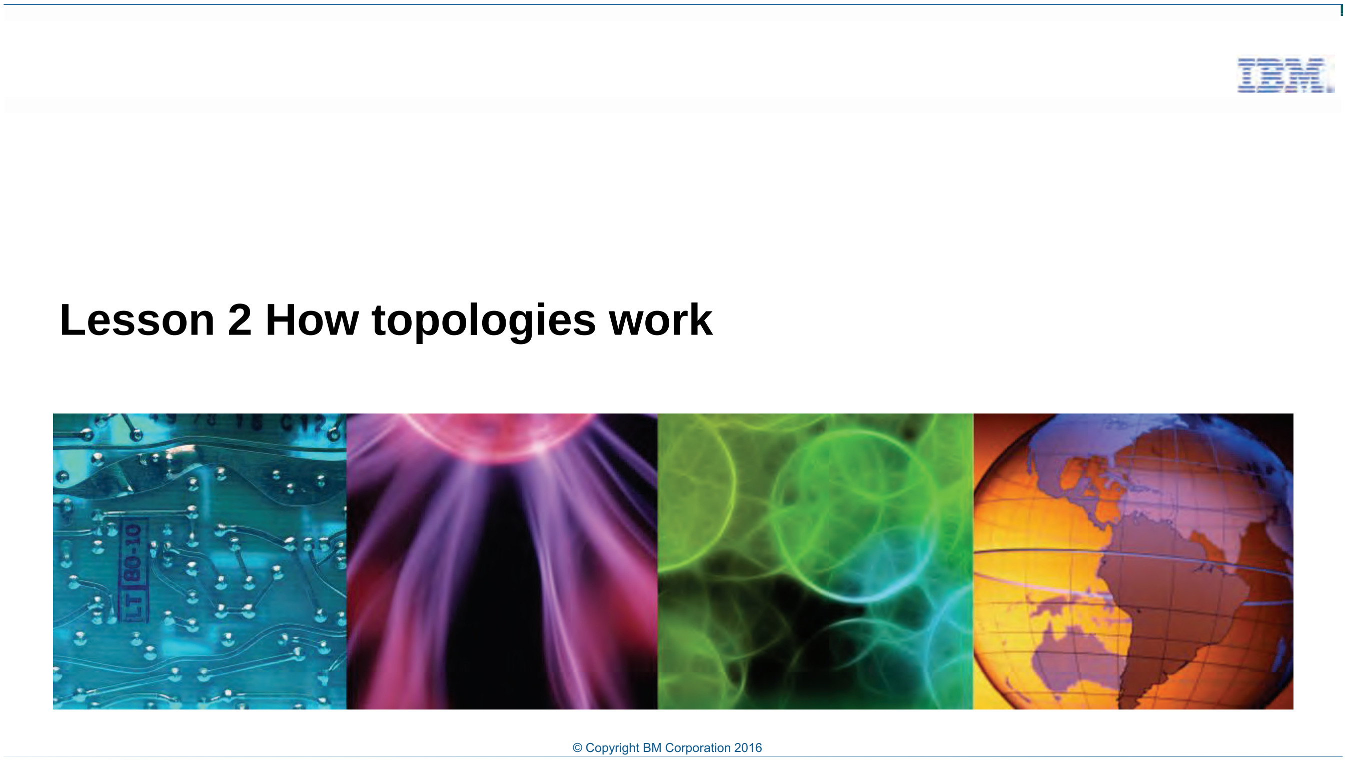
| | Packet Analyzer | Response Time Monitoring with Client Time (JavaScript Instrumentation) V8.1 and earlier | IBM HTTP Server Response Time module V8.1.1 and later |
|---|---|---|---|
| Transactions Top 10 | ✓ | ✓ | ✓ |
| Server Time | ✓ | ✓ | ✓ |
| Render Time Breakdown | ✗ | ✓ | ✓ |
| AJAX Subtransactions | ✓ | ✓ | ✓ |
| Resource Timing data in Subtransactions table | ✗ | ✓ | ✓ |
| Transaction Instances (Top 10) | ✓ | N/A | ✓ |
| Transaction Instance Topology | ✓ | ✓ | ✓ |
| Application Topology | ✓ | ✓ | ✓ |
| Automatic instrumentation of JavaScript Injection | N/A | ✗ | ✓ |

*Supported transaction features*

IBM HTTP Server Response Time module participates in transaction tracking by monitoring the web server port used by IBM HTTP Server. Use either the IBM HTTP Server Response Time module or Packet Analyzer to monitor IBM HTTP Server, not both.

The slide table shows the features available with the different monitors.

# Lesson 2  How topologies work



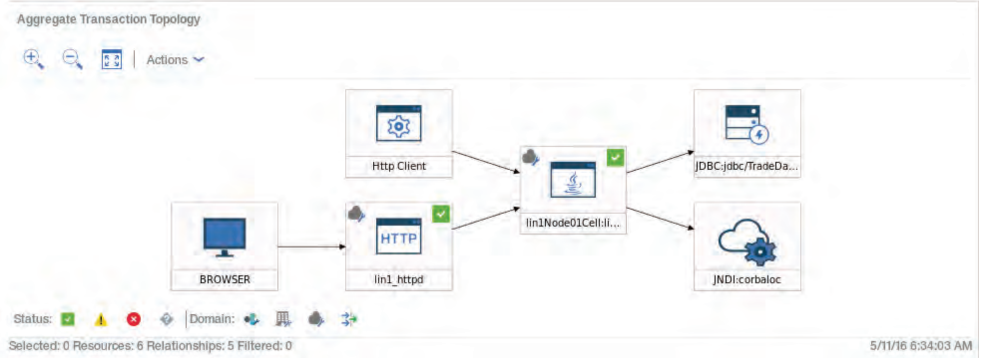
## Lesson 2 How topologies work

© Copyright BM Corporation 2016

In this lesson, you cover key features of transaction tracking.

## Transaction topology

- Text is resource name
- Status indicator:
- Green = Good
- Yellow = Slow
- Red = Failed



- Two types: aggregate and instance
- Node icon indicates resource type
- Single head: unidirectional
- Multiple-head: bidirectional
- Move the mouse pointer over a node to see tooltip.

*Transaction topology*

Each node represents the transaction data within a monitored domain. There are two types of topologies: aggregate and instance. An aggregate topology is a topology of aggregated transaction data, which is the average performance of a transaction over the aggregate interval. An instance topology is a topology based on a single transaction instance and shows actual time values.

The node image is unique for each monitored domain. Lines between nodes show transactions flowing between nodes. Lines with one arrow head indicate the transaction requests flow in one direction only. Lines with two arrow heads indicate bidirectional transactions. A status indicates for good, slow, or failed is placed on the upper right of each node.

Roll your cursor over a node for details on that node.

## Using transaction topologies

**Application transaction monitoring**
Detect which transactions (types) are contributing to a bad user experience.

Query the TopN list of transaction instances

**Problem determination**
View the TopN problematic transaction instances for this type.

Start an instance trace to correlate the end-to-end pieces of the transaction.

**Isolate bottlenecks**
Visualize where time is spent in the transaction.

Launch into code-level for the bottleneck.

**Identify root cause**
Use the method trace to identify the line of code.

*Using transaction topologies*

You can use application transaction monitoring to detect which transactions (types) are contributing to a bad user experience, for example, AccountBalance().

Query the TopN list of transaction instances, and proceed with problem determination.

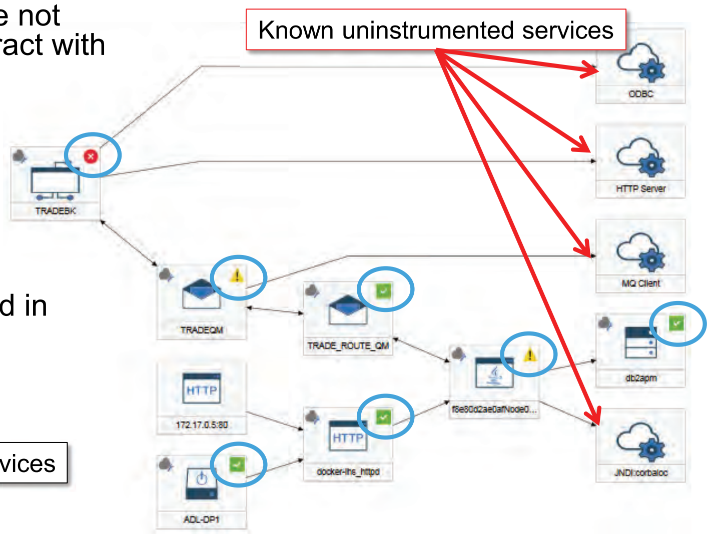
Next review the TopN problematic transaction instances for this type. Select a possibly problematic transaction and click it to launch an instance trace to correlate the end-to-end pieces of the transaction.

Isolate potential bottlenecks, and use the transaction topology to visualize where time is spent in the transaction.

Examine the deep-dive screens to identify the root cause, looking at things like request context and method trace features to identify the line of code.

## Uninstrumented Service Tracking

- Mechanism to visualize services that are not tracked by transaction tracking, but interact with tracked domains
  Referred to as pseudo nodes
- Known uninstrumented services
  Get service-specific icon
- Unknown uninstrumented services
  Get generic icon
- Display in topology enabled and disabled in Advanced Configuration
  Disabled by default



Known uninstrumented services

○ = Instrumented services

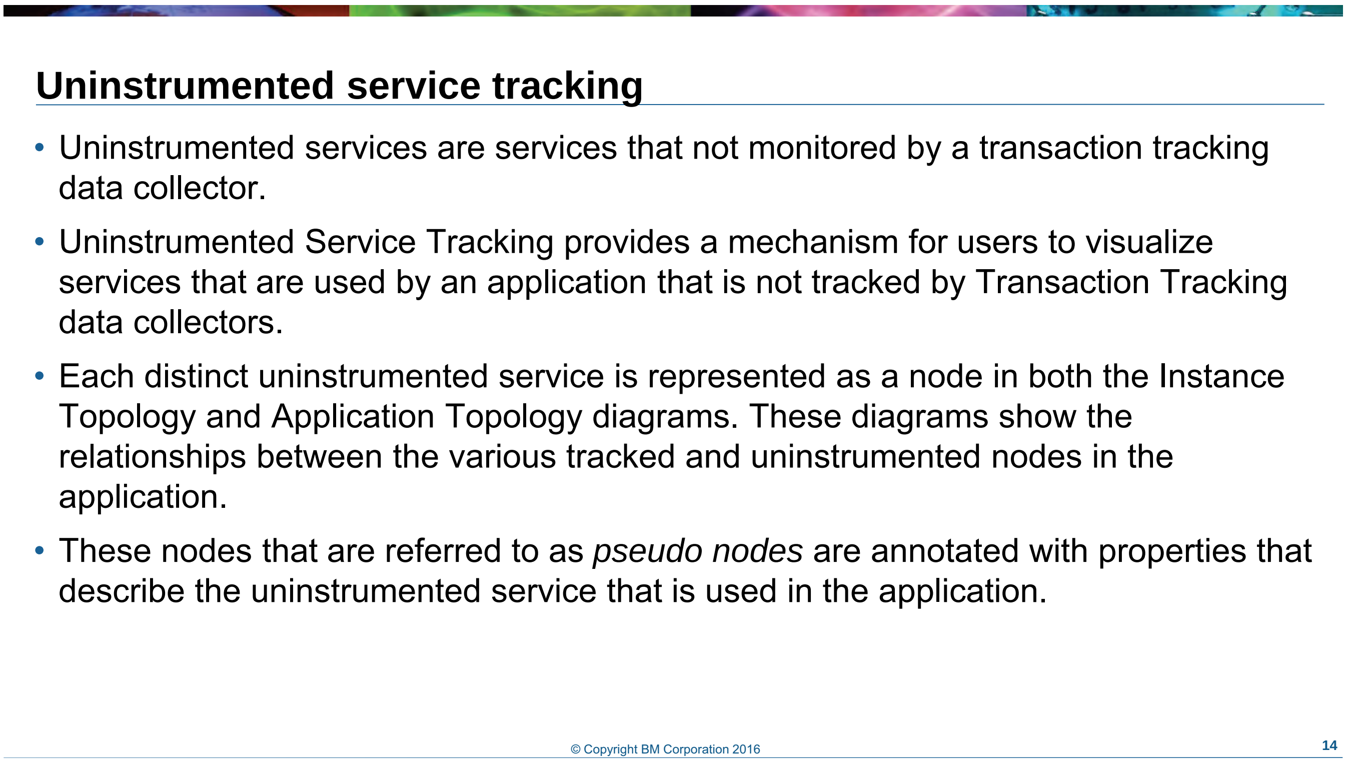
*Uninstrumented Service Tracking*

Uninstrumented Service Tracking provides a mechanism for users to visualize services that are used by an application not tracked by transaction tracking data collectors. Uninstrumented services are services that are not monitored directly by a transaction tracking agent, but either call or are called by a monitored domain.

Each distinct uninstrumented service is represented as a node in both the Instance Topology and Application Topology diagrams. These diagrams show the relationships between the various tracked and uninstrumented nodes in the application.

These nodes are referred to as *pseudo-nodes*. They are annotated with properties that describe the uninstrumented service that is used in the application.

By default, pseudo-nodes are disabled in the Performance Management console Advanced Configuration; enable pseudo-nodes to see uninstrumented services in Instance and Application Topologies.

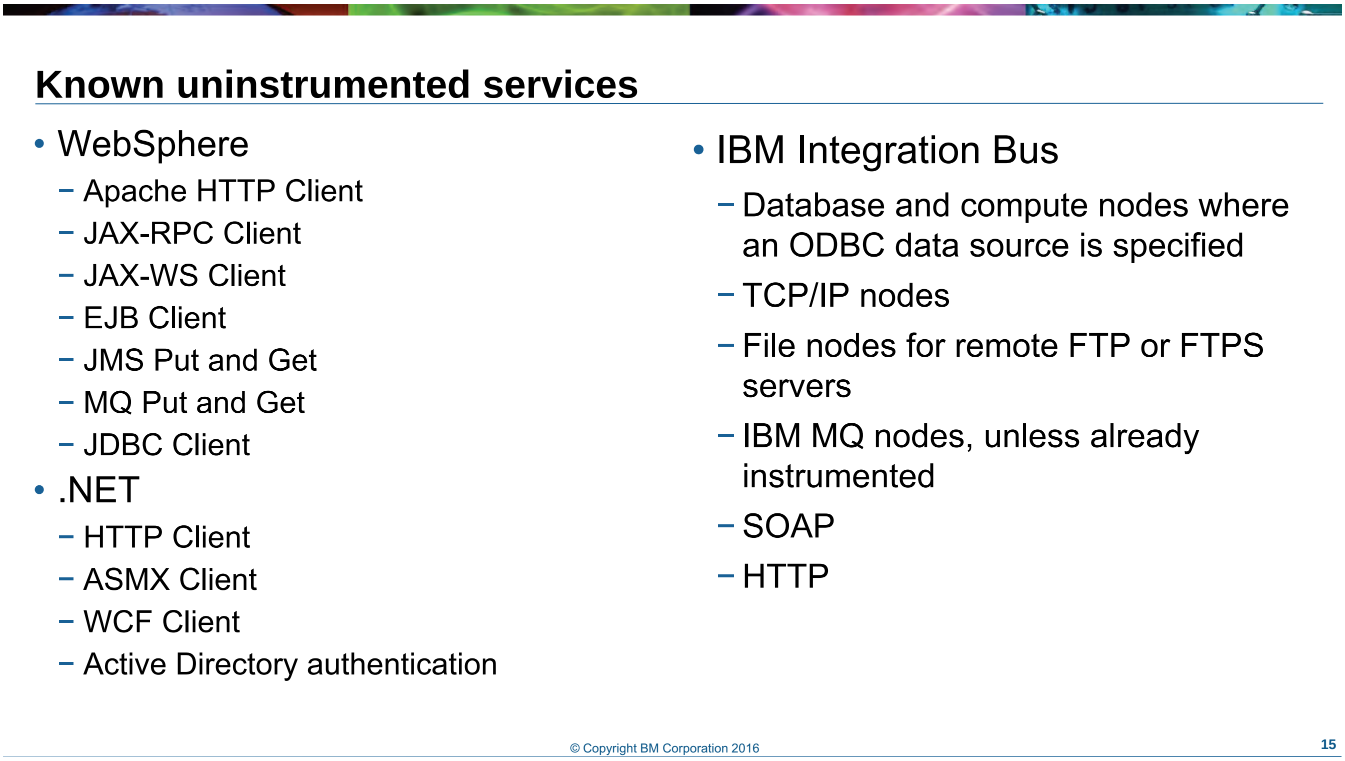


## Uninstrumented service tracking

- Uninstrumented services are services that not monitored by a transaction tracking data collector.
- Uninstrumented Service Tracking provides a mechanism for users to visualize services that are used by an application that is not tracked by Transaction Tracking data collectors.
- Each distinct uninstrumented service is represented as a node in both the Instance Topology and Application Topology diagrams. These diagrams show the relationships between the various tracked and uninstrumented nodes in the application.
- These nodes that are referred to as *pseudo nodes* are annotated with properties that describe the uninstrumented service that is used in the application.

*Uninstrumented service tracking*

Uninstrumented services are services that are not monitored by a transaction tracking data collector.Uninstrumented Service Tracking provides a mechanism for users to visualize services used by an application which are not directly tracked by Transaction Tracking data collectors.



## Known uninstrumented services

- WebSphere
  - Apache HTTP Client
  - JAX-RPC Client
  - JAX-WS Client
  - EJB Client
  - JMS Put and Get
  - MQ Put and Get
  - JDBC Client
- .NET
  - HTTP Client
  - ASMX Client
  - WCF Client
  - Active Directory authentication
- IBM Integration Bus
  - Database and compute nodes where an ODBC data source is specified
  - TCP/IP nodes
  - File nodes for remote FTP or FTPS servers
  - IBM MQ nodes, unless already instrumented
  - SOAP
  - HTTP

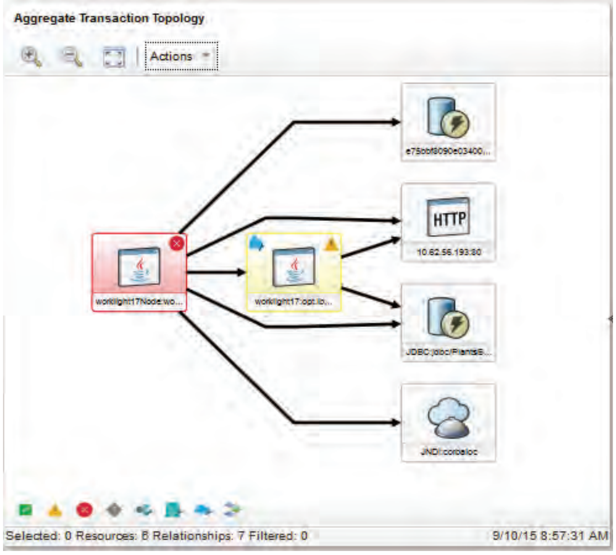
*Known uninstrumented services*

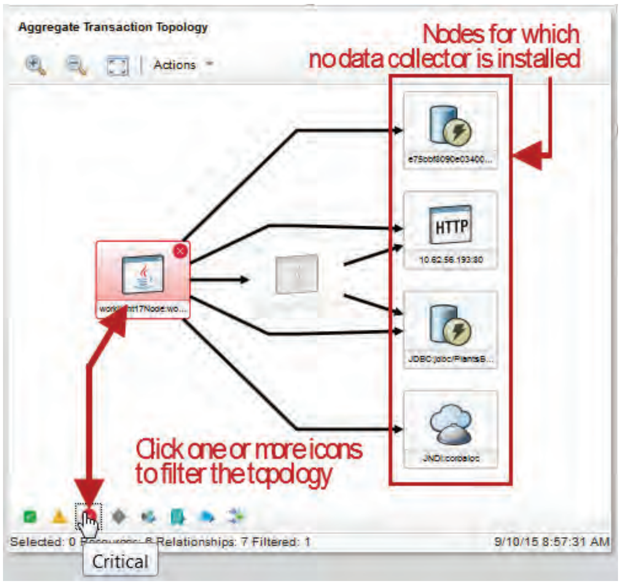
A typical scenario incorporates a browser that communicates with an instrumented app server, which, in turn, communicates with an uninstrumented App Server.

Supported scenarios, by application server, include these examples:

- A topology for a transaction originating with a WebSphere application server can show pseudo-nodes for uninstrumented services including these examples:
  - Apache HTTP Client
  - JAX-RPC Client
  - JAX-WS Client
  - EJB Client
  - JMS Put and Get
  - MQ Put and Get
- A topology for a transaction originating with a .NET application server can show pseudo-nodes for uninstrumented services, including these examples:
  - HTTP Client
  - ASMX Client
  - WCF Client
  - Active Directory authentication

## Filtering application topologies





Nodes for which no data collector is installed

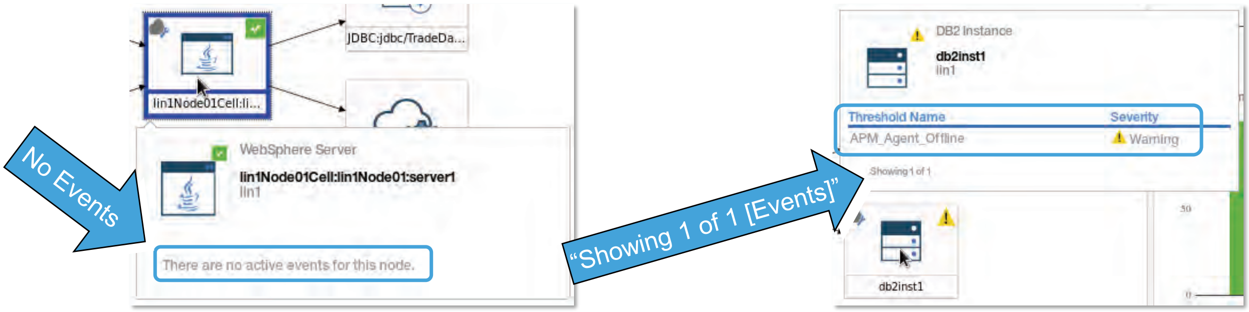
Click one or more icons to filter the topology

*Filtering application topologies*

You can filter application topologies using the icons at the bottom left of the topology window.The topology on the left is typical of a transaction that makes a call to an instrumented application server. The transaction also includes a web service on an uninstrumented application server, the bottom node on the right.

In the topology on the right, the Critical node is selected. The WebSphere node in the middle of the topology is gray and unavailable in the topology because it is not in a critical state. The four nodes at the right are still visible because no data collector is installed. The nodes are therefore stateless and do not fit within any of the categories represented by the filtering icons.

## Topology tooltip



No Events

WebSphere Server
**lin1Node01Cell:lin1Node01:server1**
lin1

There are no active events for this node.

"Showing 1 of 1 [Events]"

DB2 Instance
**db2inst1**
lin1

| Threshold Name | Severity |
|---|---|
| APM_Agent_Offline | ⚠ Warning |

Showing 1 of 1

Events

*Topology tooltip*

Roll your cursor over a topology node to see the tooltip.

The tooltip shows this information:

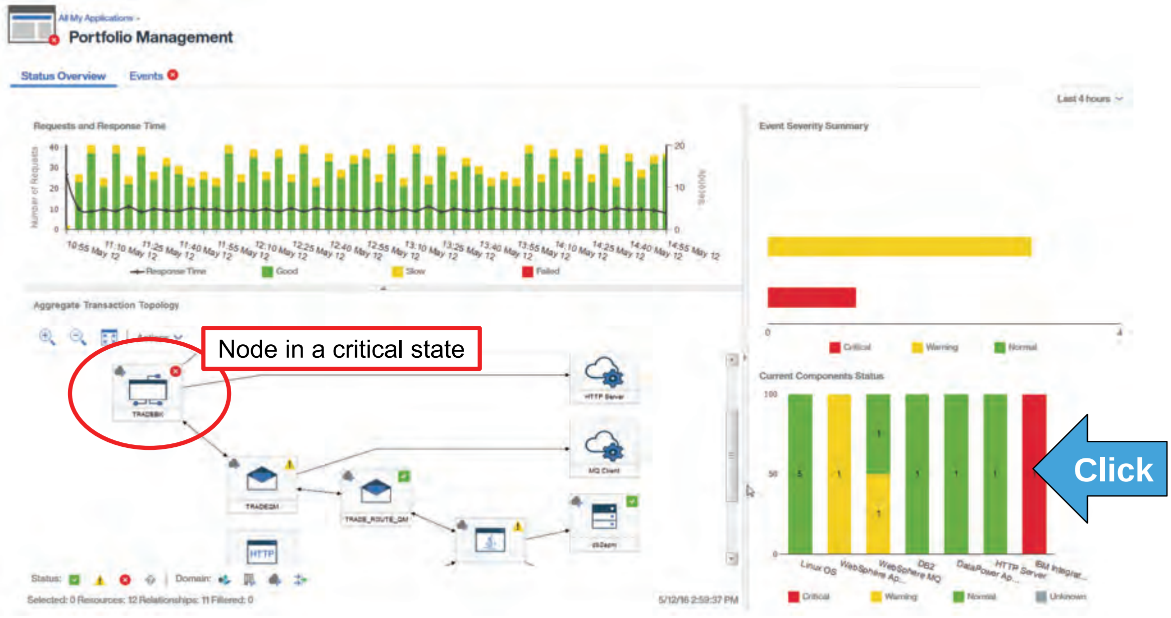
- Resource type image
- Status image
- Resource type
- Resource name
- Host name
- List of events, if any, or a or a message indicating there are no events
- Number of events showing and the total number of events

# Lesson 3  Using topologies

## Lesson 3 Using topologies

In this lesson, you discover how to use and interpret transaction topologies.

## Aggregate topology



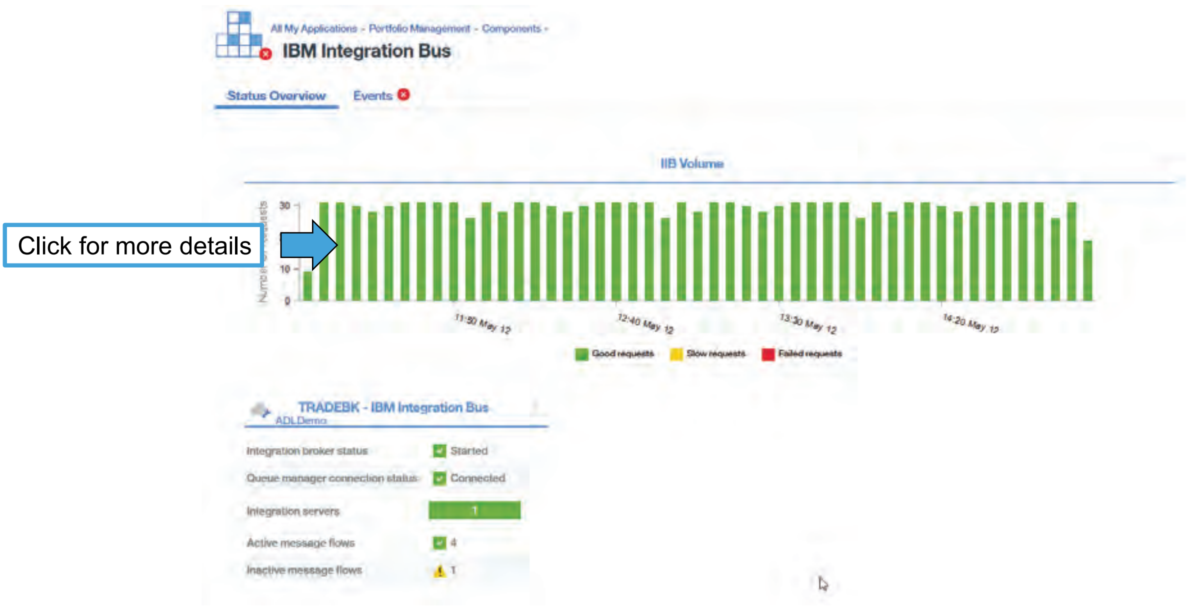
Node in a critical state

Click

*Aggregate topology*

When transaction tracking is enabled on any monitoring component within an application, an Aggregate Transaction topology widget is displayed on the Application Summary page. The Critical icon is for the instance of IBM Integration Bus (IIB). Click the corresponding bar in the Current Components Status widget to access the Component page for IIB.

## IIB transaction volume



All My Applications - Portfolio Management - Components -

IBM Integration Bus

Status Overview | Events

TRADEBK - IBM Integration Bus
ADLDemo

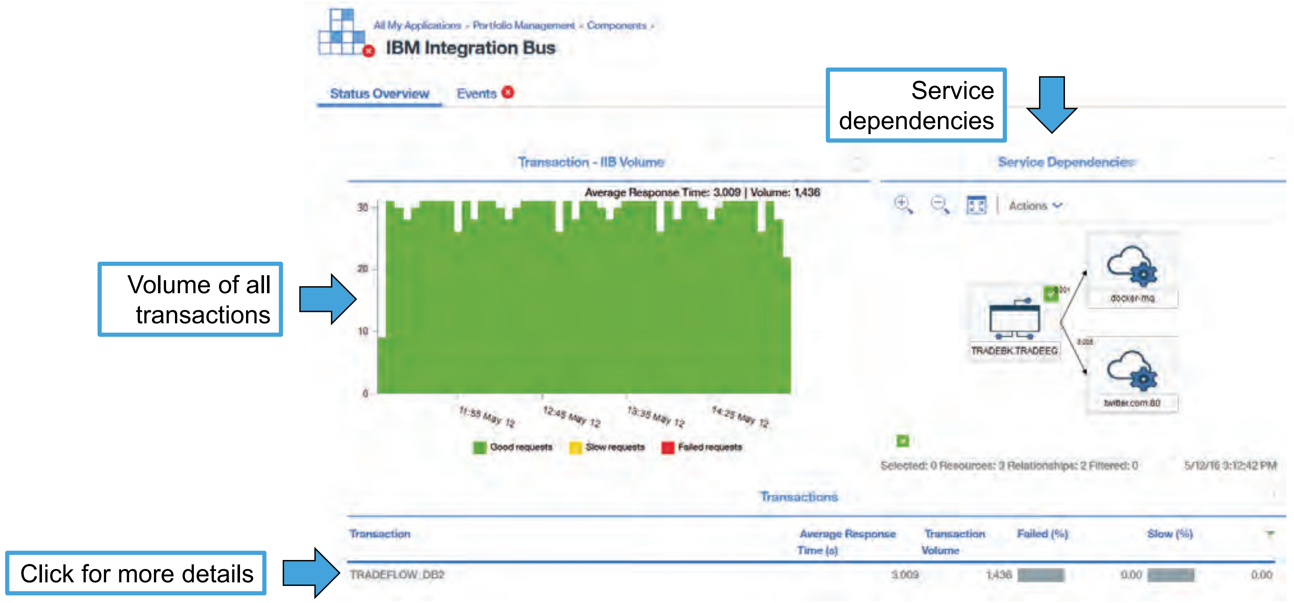
| | |
|---|---|
| Integration broker status | ✓ Started |
| Queue manager connection status | ✓ Connected |
| Integration servers | 1 |
| Active message flows | ✓ 4 |
| Inactive message flows | ⚠ 1 |

Click for more details

*IIB transaction volume*

The Transactions Volume widget shows the numbers of transactions received in each aggregation interval. For each interval bar, color coding shows the percent of transactions that were good, slow, or failed. Click the IIB Volume widget for Middleware transaction details.

## Middleware Transaction Details



| Transaction | Average Response Time (s) | Transaction Volume | Failed (%) | Slow (%) |
|---|---|---|---|---|
| TRADEFLOW_DB2 | 3.009 | 1,436 | 0.00 | 0.00 |

Service dependencies

Volume of all transactions

Click for more details

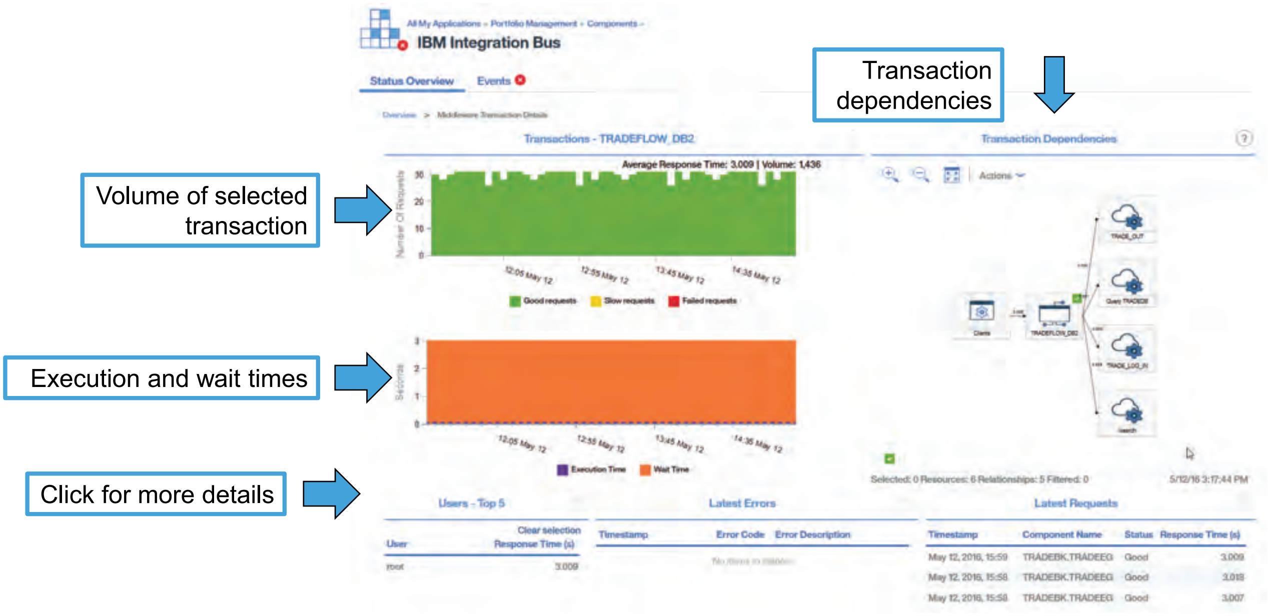
© Copyright BM Corporation 2016

21

*Middleware Transaction Details*

The Middleware Transaction Details page shows data over time for a specific transaction and includes the widgets shown in the slide. A Service Dependencies widget is an addition in version 8.1.3. Click an entry in the Transactions widget for additional detail, including Transaction dependencies.

## Middleware Transaction Details (Cont.)



All My Applications > Portfolio Management > Components >

IBM Integration Bus

Status Overview | Events

Overview > Middleware Transaction Details

Transactions - TRADEFLOW_DB2

Average Response Time: 3,009 | Volume: 1,436

Transaction Dependencies

Selected: 0 Resources: 6 Relationships: 5 Filtered: 0    5/12/16 3:17:44 PM

Users - Top 5

Clear selection

| User | Response Time (s) |
|---|---|
| root | 3.009 |

Latest Errors

| Timestamp | Error Code | Error Description |
|---|---|---|

No items to display

Latest Requests

| Timestamp | Component Name | Status | Response Time (s) |
|---|---|---|---|
| May 12, 2016, 15:59 | TRADEBK.TRADEEG | Good | 3.009 |
| May 12, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.018 |
| May 12, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.007 |

Volume of selected transaction

Execution and wait times

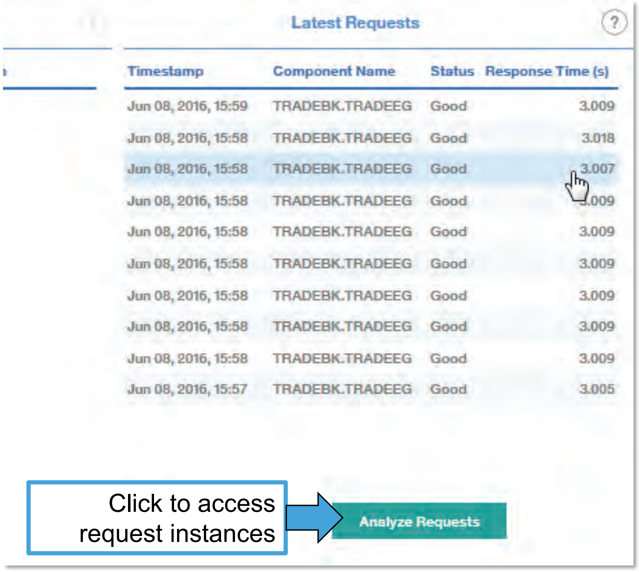
Click for more details

Transaction dependencies

*Middleware Transaction Details (Cont.)*

At this next level, Middleware Transaction Details shows data over time for a specific transaction and includes the widgets shown in the slide. A Transaction Dependencies widget is an addition in version 8.1.3. The other widgets at this layer display data of users, latest errors, and latest requests.

## Middleware Transaction Details (Cont.)



Latest Requests

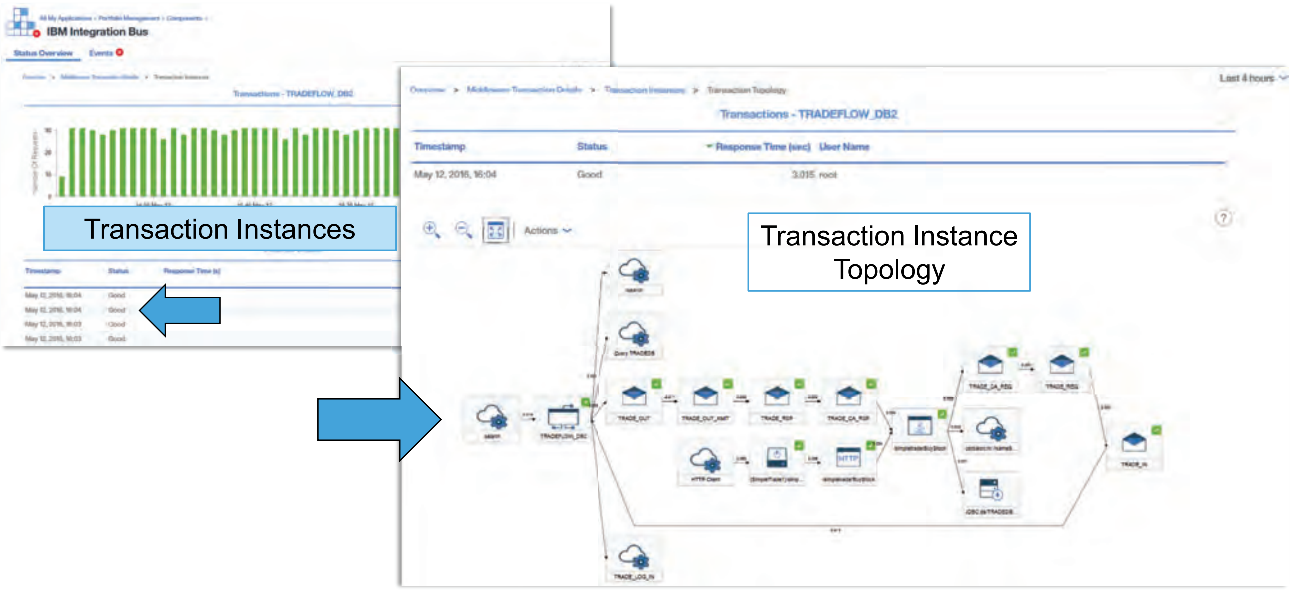
| Timestamp | Component Name | Status | Response Time (s) |
|---|---|---|---|
| Jun 08, 2016, 15:59 | TRADEBK.TRADEEG | Good | 3.009 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.018 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.007 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.009 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.009 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.009 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.009 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.009 |
| Jun 08, 2016, 15:58 | TRADEBK.TRADEEG | Good | 3.009 |
| Jun 08, 2016, 15:57 | TRADEBK.TRADEEG | Good | 3.005 |

Click to access request instances

*Middleware Transaction Details (Cont.)*

At this next level, click **Analyze Requests** to view request instances.

## Middleware Transaction Instances



IBM Integration Bus

Status Overview | Events

Transactions - TRADEFLOW_DB2

Transaction Instances

| Timestamp | Status | Response Time (s) |
|---|---|---|
| May 12, 2016, 16:04 | Good | |
| May 12, 2016, 16:04 | Good | |
| May 12, 2016, 16:03 | Good | |
| May 12, 2016, 16:03 | Good | |

Overview > Middleware Transaction Details > Transaction Instances > Transaction Topology

Last 4 hours

Transactions - TRADEFLOW_DB2

| Timestamp | Status | Response Time (sec) | User Name |
|---|---|---|---|
| May 12, 2016, 16:04 | Good | 3.015 | root |

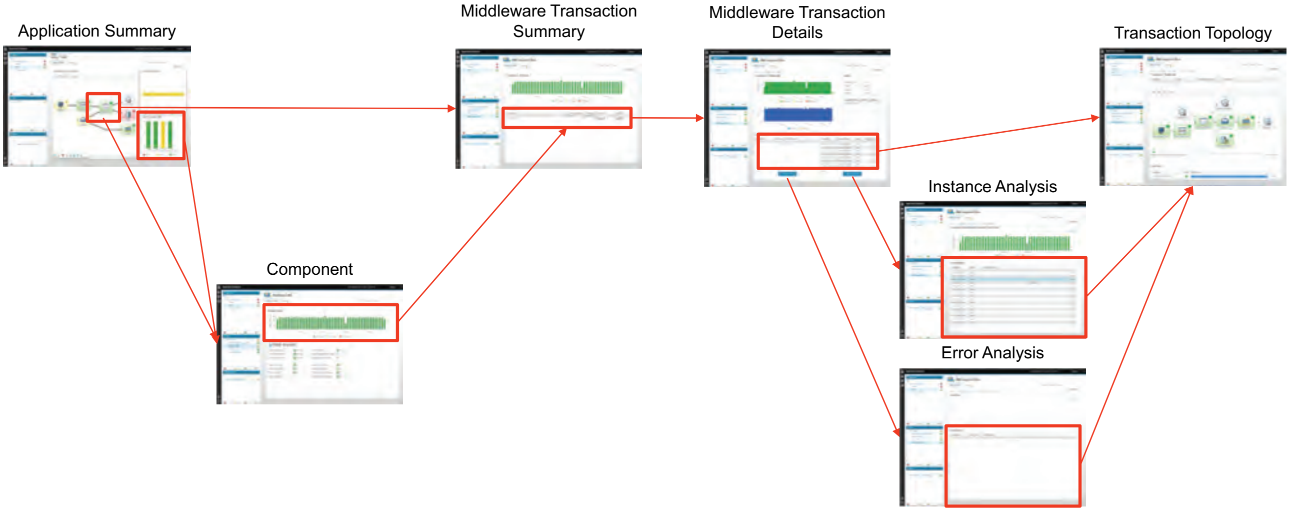
Transaction Instance Topology

24

*Middleware Transaction Instances*

Transaction instance details are available, including the status and response time of each instance. Click an instance to go to the Instance Topology page.

## Navigating transaction tracking



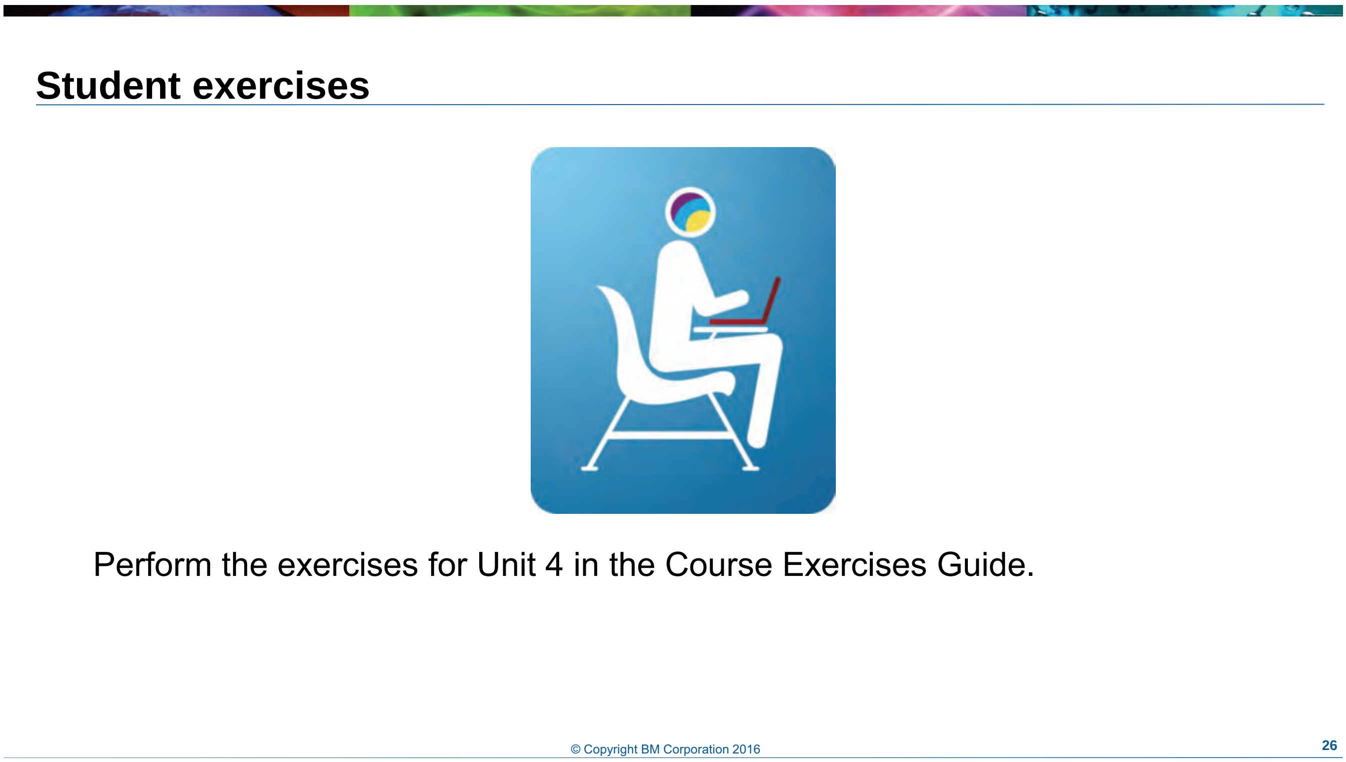
*Navigating transaction tracking*

To drill down from Aggregate Transaction topology, right-click a middleware node and select **Go to Transaction Summary page**.

To drill down from the Component page, click the Transactions widget.

To drill down to an instance topology page, click the instance in either the Middleware Transaction Details, Instance Analysis, or Error Analysis page.

To drill down to the analysis pages, click the appropriate **Analyse** button on the Middleware Transaction Details page.



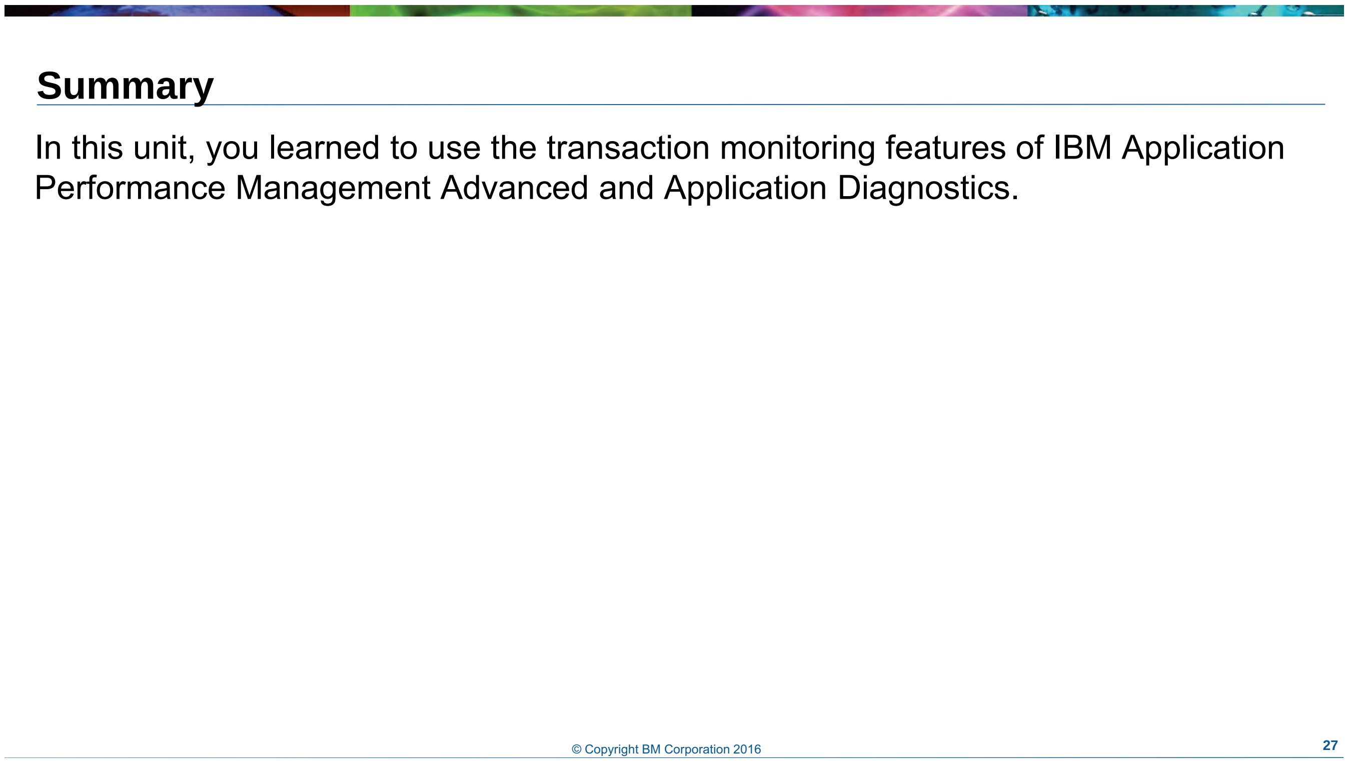
## Student exercises

Perform the exercises for Unit 4 in the Course Exercises Guide.

26

*Student exercises*

You now perform the exercises for Unit 4 in the Course Exercises Guide.



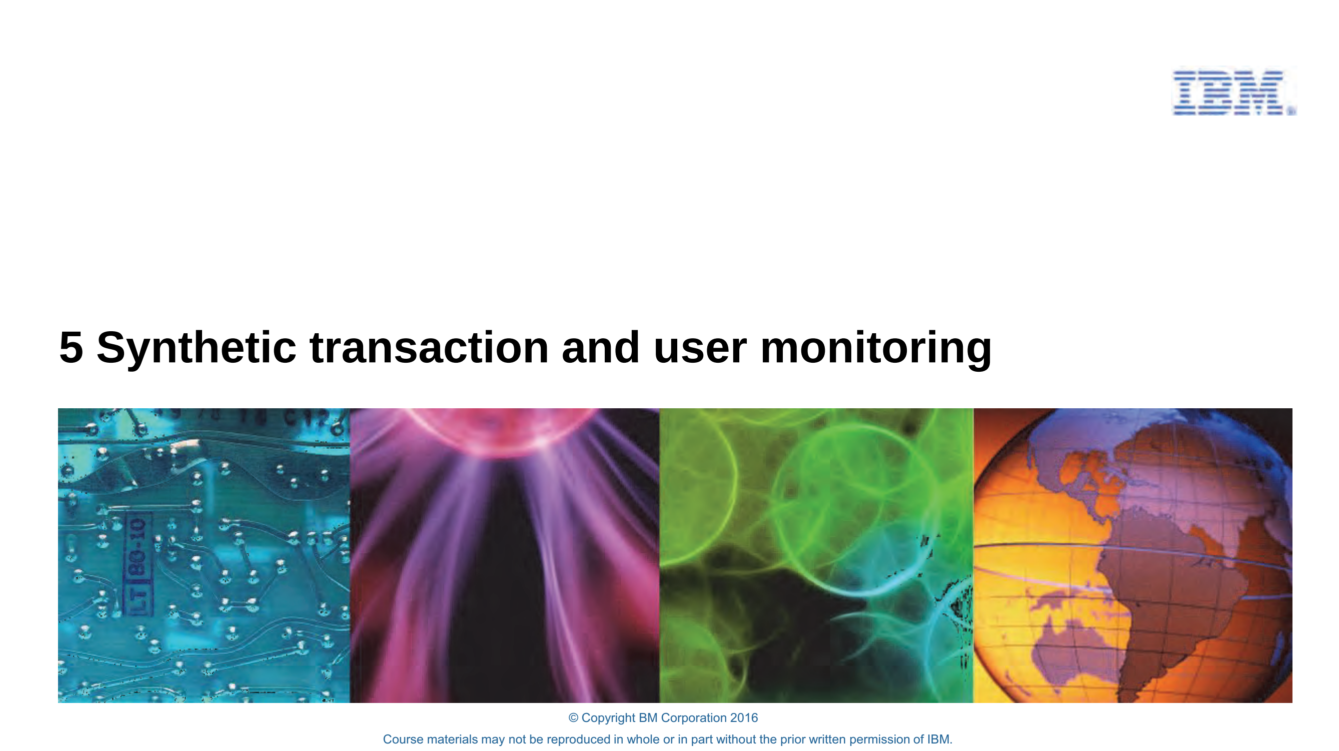
## Summary

In this unit, you learned to use the transaction monitoring features of IBM Application Performance Management Advanced and Application Diagnostics.

*Summary*

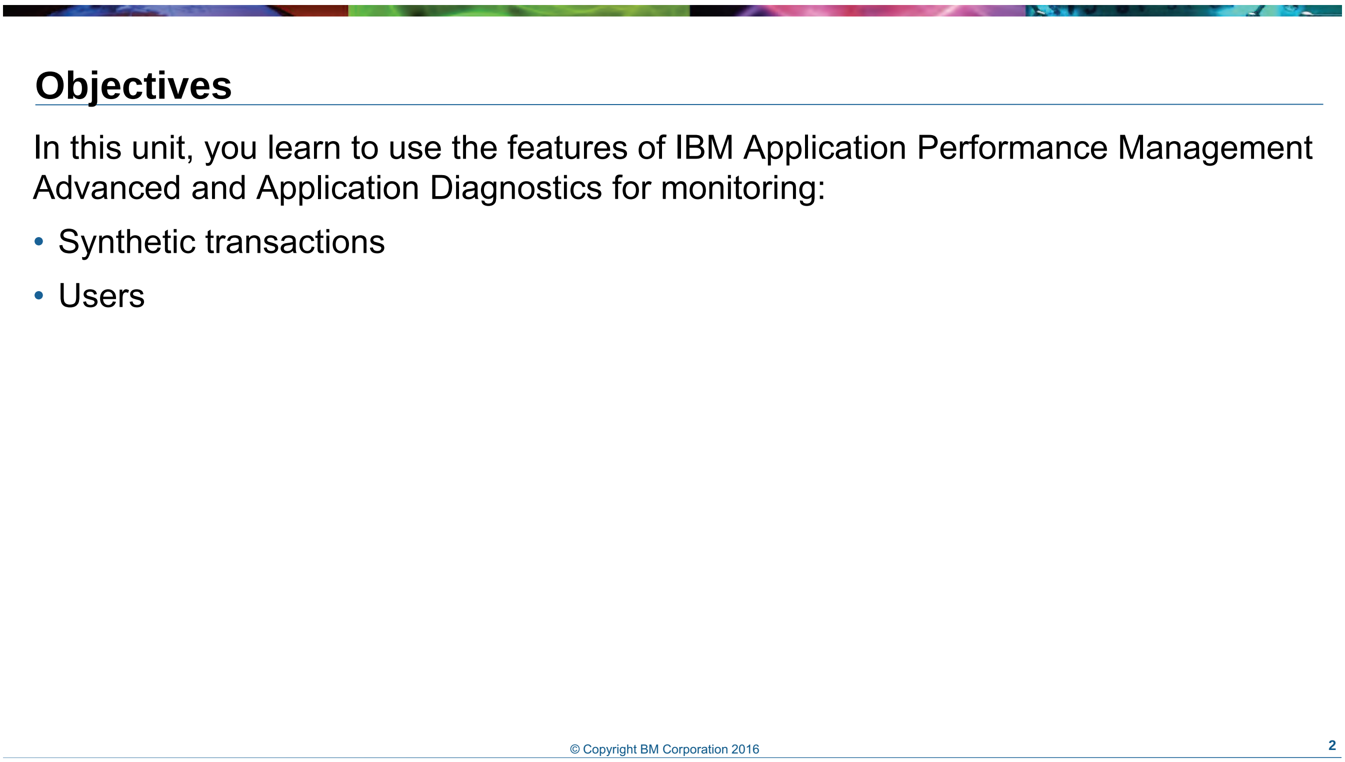
In this unit, you learned to describe the transaction monitoring features of IBM Application Performance Management Advanced and Application Diagnostics.

# 5 Synthetic transaction and user monitoring



## 5 Synthetic transaction and user monitoring

This presentation is an overview of the transaction monitoring features of IBM Application Performance Management Advanced for synthetic transactions, websites, and users.



## Objectives

In this unit, you learn to use the features of IBM Application Performance Management Advanced and Application Diagnostics for monitoring:

- Synthetic transactions
- Users

© Copyright BM Corporation 2016

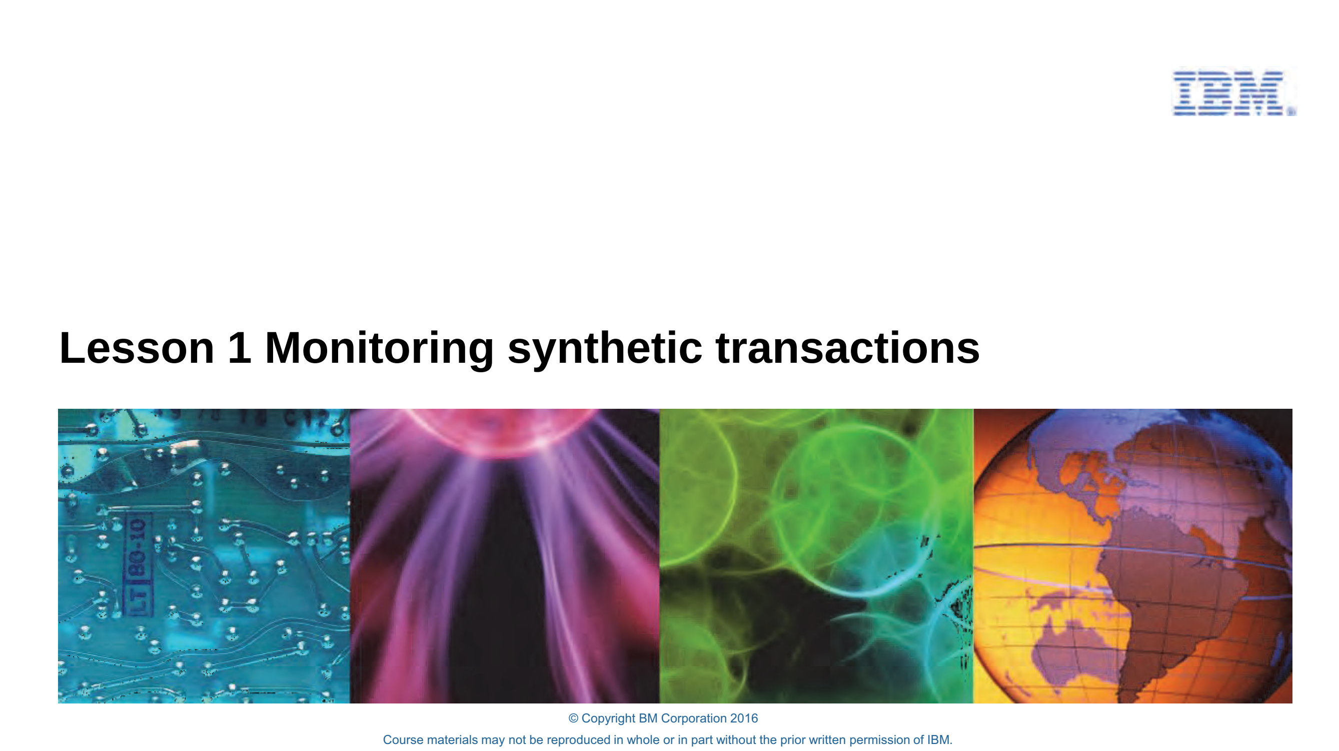
2

*Objectives*

In this unit, you learn to use the features of IBM Application Performance Management Advanced and Application Diagnostics for monitoring these items:

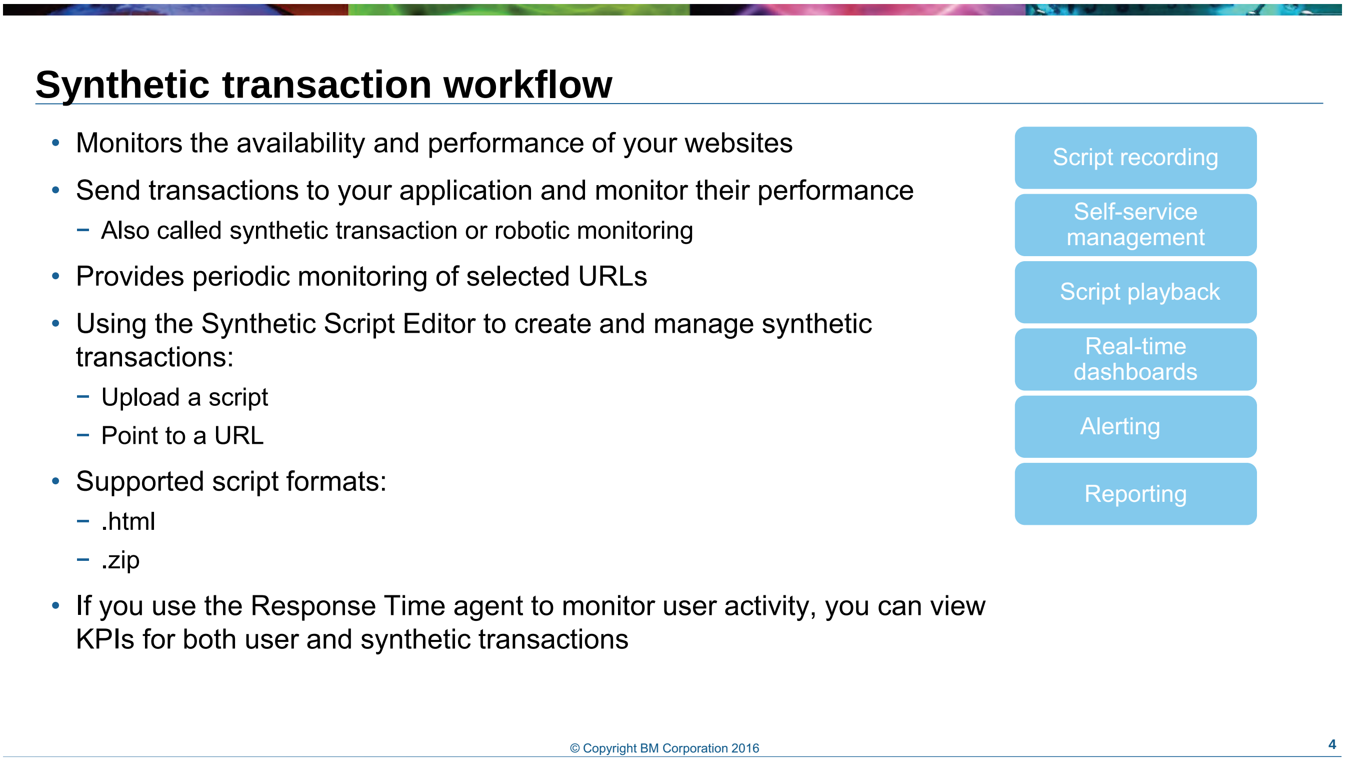
- Synthetic transactions
- Users

# Lesson 1  Monitoring synthetic transactions



## Lesson 1 Monitoring synthetic transactions

In the first section of this presentation, you get an overview of the how IBM Performance Management Advanced monitors synthetic transactions.



## Synthetic transaction workflow

- Monitors the availability and performance of your websites
- Send transactions to your application and monitor their performance
  - Also called synthetic transaction or robotic monitoring
- Provides periodic monitoring of selected URLs
- Using the Synthetic Script Editor to create and manage synthetic transactions:
  - Upload a script
  - Point to a URL
- Supported script formats:
  - .html
  - .zip
- If you use the Response Time agent to monitor user activity, you can view KPIs for both user and synthetic transactions

Script recording

Self-service management

Script playback

Real-time dashboards

Alerting

Reporting

© Copyright BM Corporation 2016

4

*Synthetic transaction workflow*

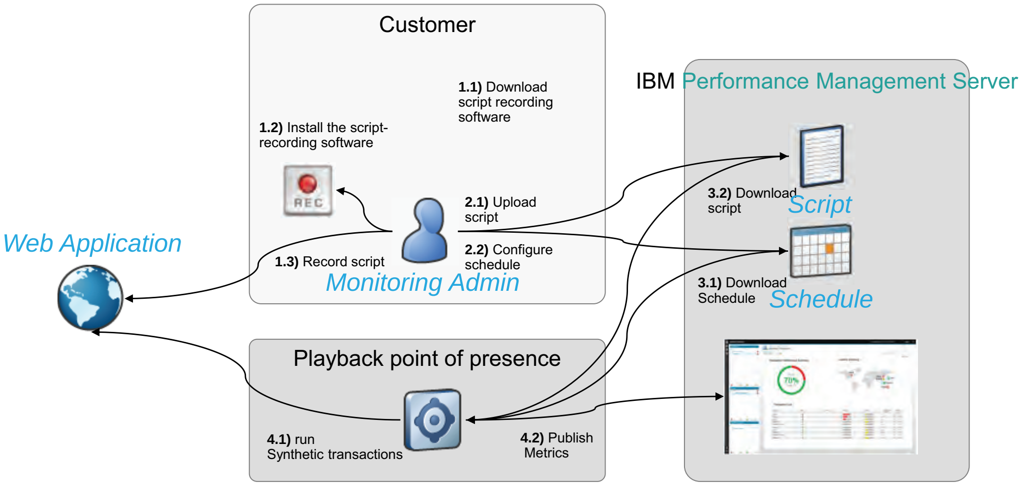
Monitoring with synthetic transactions provides guaranteed, periodic monitoring of the performance and availability of your websites from multiple geographically distributed points of presence.

Create a synthetic transaction in the Synthetic Script Manager. Generate simple scripts in the Synthetic Script Manager to test the availability of an application, or use Selenium-IDE to record synthetic scripts that replicate different user actions with an application. Then configure a synthetic transaction to play back your script at specific intervals and playback locations.

Create thresholds and resource groups to raise events and notify stakeholders when your applications are slow or unavailable. View performance data and generate historical reports in the Application Performance Dashboard.

If you monitor user response time for an application with the Response Time agent, you can view KPIs for both user and synthetic transactions in the Application Performance Dashboard. Add synthetic transactions as components to the application that you are monitoring with the Response Time agent.

## Synthetic monitoring workflow (Cont.)



© Copyright BM Corporation 2016

*Synthetic monitoring workflow (Cont.)*

This slide shows the synthetic monitoring workflow. The customer identifies a single URL to monitor or records a script that tests multiple website URLs.

The user then creates the synthetic transaction in the PM Console, entering the single URL or uploading the recorded script. The synthetic transaction has a playback schedule, points of presence, and thresholds.

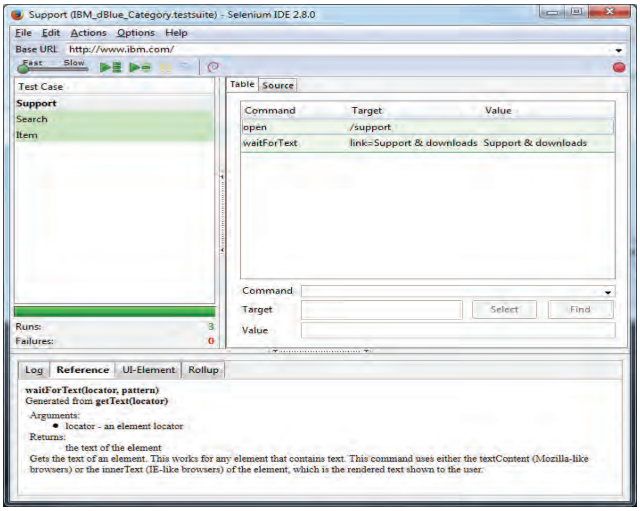
With the cloud solution (WebSite Monitoring), IBM provides different points of presence around the world. With WebSite Monitoring, the cloud user can optionally create their own points of presence by deploying the Synthetic Playback agent. With the on premises solution, users must create their own points of presence by deploying the Synthetic Playback agent.

The script plays back on the schedule at the identified points of presence and the synthetic transactions are monitored.

## Script recording

- Record your synthetic transactions visually by using a Firefox browser plug-in

  Download Selenium IDE from http://docs.seleniumhq.org/download/
- Built upon industry-leading Selenium technology
- Selenese scripting language provides rich capabilities for validating page contents
- Organize your scripts into multiple test cases to indicate the steps in a business process
- Download the Selenium IDE to try it out



Base URL http://www.ibm.com/

| Test Case |
|---|
| **Support** |
| Search |
| Item |

Runs: 3
Failures: 0

| Command | Target | Value |
|---|---|---|
| open | /support | |
| waitForText | link=Support & downloads | Support & downloads |

Command
Target
Value

**waitForText(locator, pattern)**
Generated from **getText(locator)**
Arguments:
- locator - an element locator

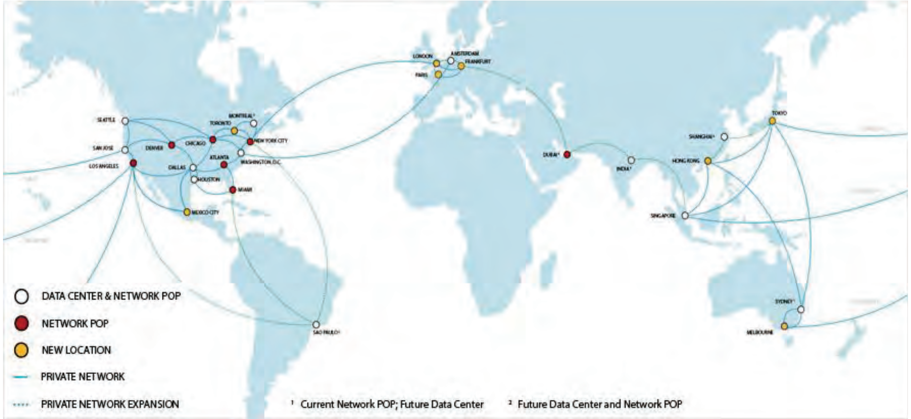
Returns:
the text of the element
Gets the text of an element. This works for any element that contains text. This command uses either the textContent (Mozilla-like browsers) or the innerText (IE-like browsers) of the element, which is the rendered text shown to the user.

*Script recording*

Record a script by using the Firefox web browser and the Selenium-IDE add-on. With Selenium-IDE, you can record user actions on a web page, such as loading a page, clicking a link, or selecting an object. When Selenium-IDE is recording, it generates a command for each user action in a script. Then, using Synthetic Script Manager, you can configure scripts to simulate user behavior at your website, at set intervals and at different locations.

## Website Monitoring points of presence (CO)

- No work on your part to create points of presence
- Website Monitoring, purchased as part of APM on Cloud, provides geographically distributed points-of-presence hosted in IBM SoftLayer data centers
- GA release locations
  - Dallas
  - Washington
  - Singapore
  - Amsterdam
  - San Jose
  - Hong Kong
  - Toronto
  - Melbourne



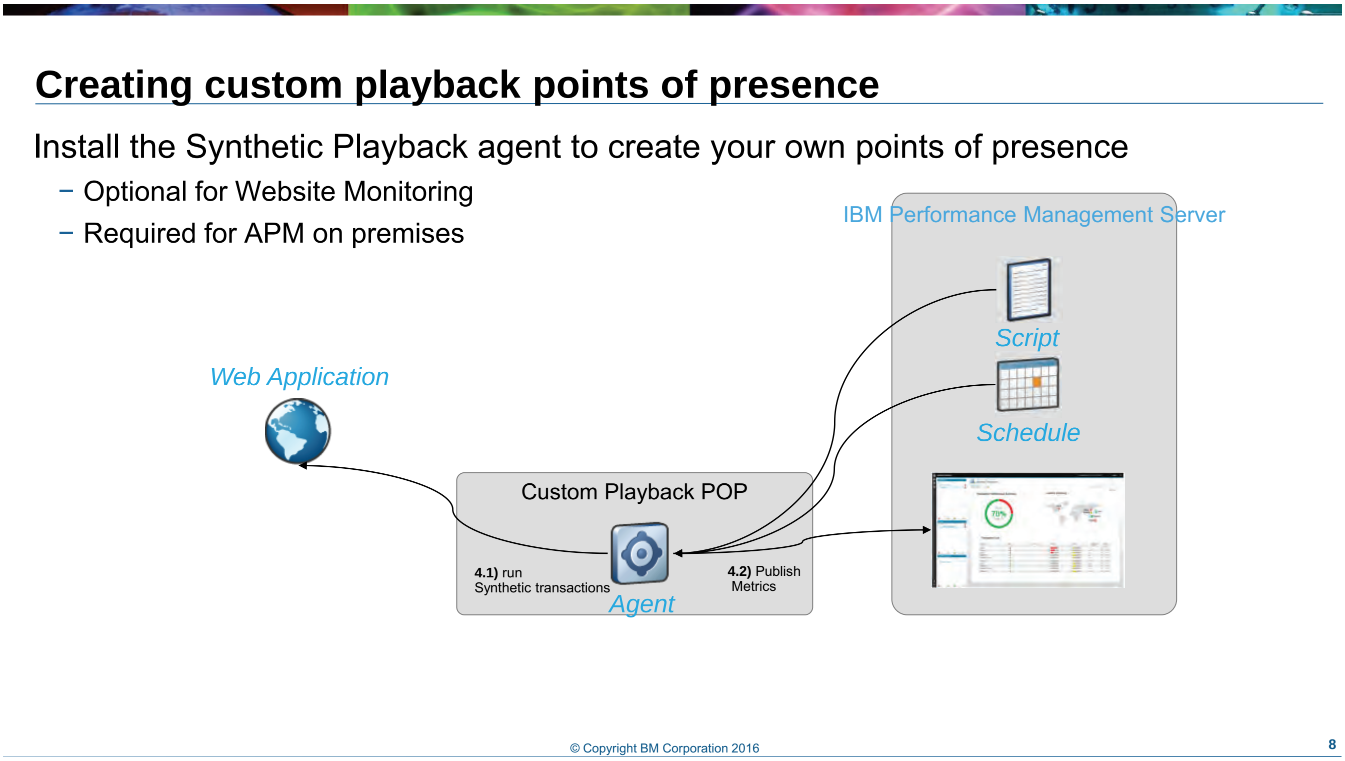
© Copyright BM Corporation 2016

7

*Website Monitoring points of presence (CO)*

IBM Website Monitoring is now available as built-in feature in the Application Performance Management and Application Performance Management Advanced offerings. You record synthetic transactions for your key business transactions and schedule them to be periodically executed from the IBM points of presence.

After a script has been uploaded and a corresponding synthetic transaction has been defined in the Synthetic Script Manager, the script automatically deploys to the global locations set in the synthetic transaction definition and playback periodically based on the defined schedule.



## Creating custom playback points of presence

Install the Synthetic Playback agent to create your own points of presence

- Optional for Website Monitoring
- Required for APM on premises

8

*Creating custom playback points of presence*

To manage synthetic transactions and events, create synthetic transactions that monitor the performance and availability of private internal-facing web applications. In the Synthetic Script Manager, either:

1. Select Enter the URL of a web page to test and enter a URL.

**Note:** The Synthetic Script Manager generates a simple synthetic script based on that URL.

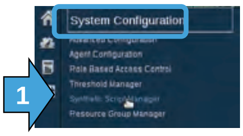
or

2. Use Selenium-IDE to record a synthetic script that replicates different user actions with an application. Import the script.

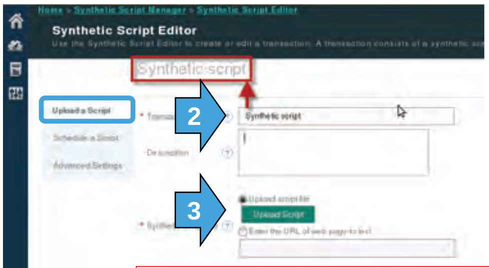
   then

3. Configure a synthetic transaction to play back your script at specific intervals and playback locations.
4. If desired, create thresholds and resource groups to raise events and notify stakeholders when your applications are slow or unavailable.
5. View performance data and generate historical reports in the Application Performance Dashboard.

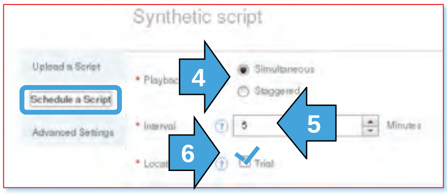
## Creating synthetic transactions



1. In the **System Configuration** menu, select the **Synthetic Script Editor**.


Home > Synthetic Script Manager > Synthetic Script Editor
Synthetic Script Editor
Use the Synthetic Script Editor to create or edit a transaction. A transaction consists of a synthetic scr
Synthetic script
Upload a Script
Schedule a Script
Advanced Settings
2
* Transa
Synthetic script
Description
3
Upload script file
Upload Script
* Synthe
Enter the URL of web page to test

2. On the **Upload a Script** tab, assign a transaction name. The name then appears at the top of the screen.
3. Select a transaction option:
   a. Upload a script.
   b. Specify a URL.

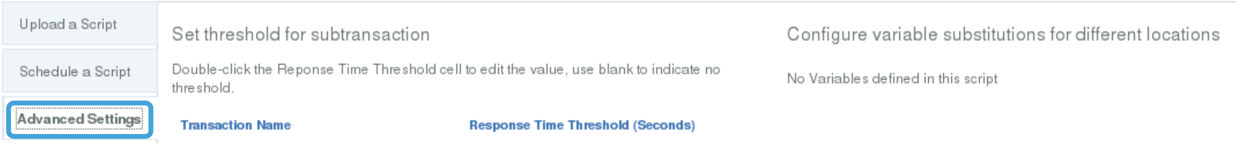


4. On the **Schedule a Script** tab, select a Playback mode:
   a. Simultaneous
   b. Staggered
5. Select a Playback interval. Default is 5 minutes.
6. Select the location that you created on configuration.

*Creating synthetic transactions*

**Note:** For Step 3: Select **Simultaneous** to execute the transaction from all locations simultaneously, or select **Staggered** to execute the transaction from a different location at each interval.

## Creating synthetic transactions (Cont.)



Set threshold for subtransaction

Double-click the Reponse Time Threshold cell to edit the value, use blank to indicate no threshold.

| Transaction Name | Response Time Threshold (Seconds) |
|---|---|

Configure variable substitutions for different locations

No Variables defined in this script

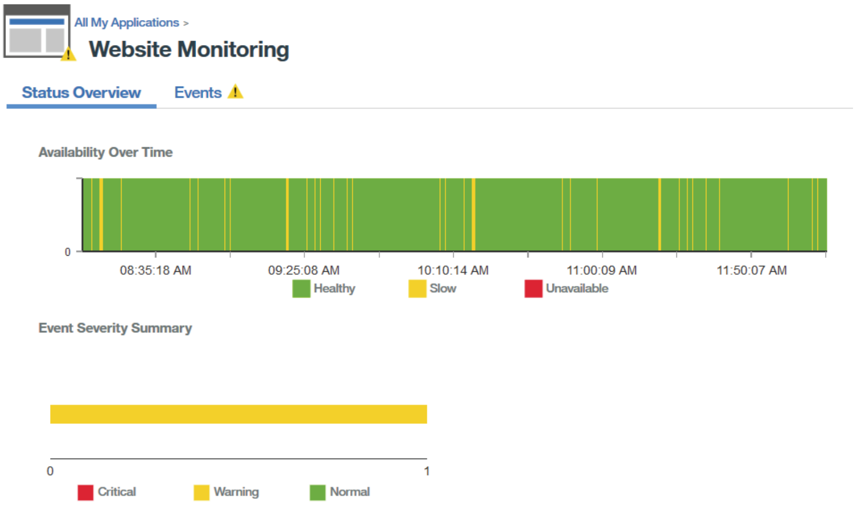
7. Where applicable, make these settings from the **Advanced Settings** tab:
   a. Set thresholds for subtransactions
   b. Configure variable substitutions
8. Save the transaction.

*Creating synthetic transactions (Cont.)*

From the **Advanced Settings** tab, you can configure subtransactions and variable substitution.

## Monitoring synthetic transactions

- Dashboards integrate across multiple IBM APM data sources
- Group scripts together under business applications
- Analyze performance by script, step, and Playback location
- Drill into a transaction instance to analyze errors
- Gantt chart visualization helps pinpoint where time is spent

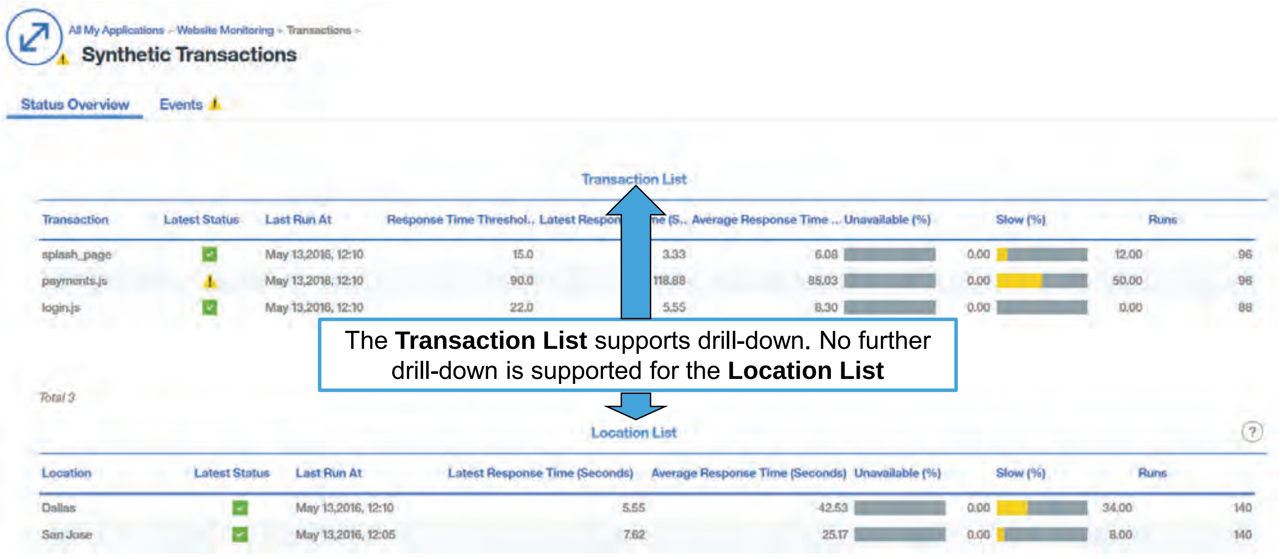


11

*Monitoring synthetic transactions*

Use the Application Performance Dashboard to add synthetic transactions as components to a new or existing web application. You can then monitor synthetic transaction data about the availability and performance of your application in the Application Performance Dashboard.

# Monitoring synthetic transactions (Cont.)



All My Applications › Website Monitoring › Transactions ›

Synthetic Transactions

Status Overview   Events

Transaction List

| Transaction | Latest Status | Last Run At | Response Time Threshol.. | Latest Respon ne (S.. | Average Response Time ... | Unavailable (%) | Slow (%) | Runs |
|---|---|---|---|---|---|---|---|---|
| splash_page | ✓ | May 13,2016, 12:10 | 15.0 | 3.33 | 6.08 | 0.00 | 12.00 | 96 |
| payments.js | ⚠ | May 13,2016, 12:10 | 90.0 | 118.88 | 85.03 | 0.00 | 50.00 | 96 |
| login.js | ✓ | May 13,2016, 12:10 | 22.0 | 5.55 | 8.30 | 0.00 | 0.00 | 88 |

*Total 3*

The **Transaction List** supports drill-down. No further drill-down is supported for the **Location List**

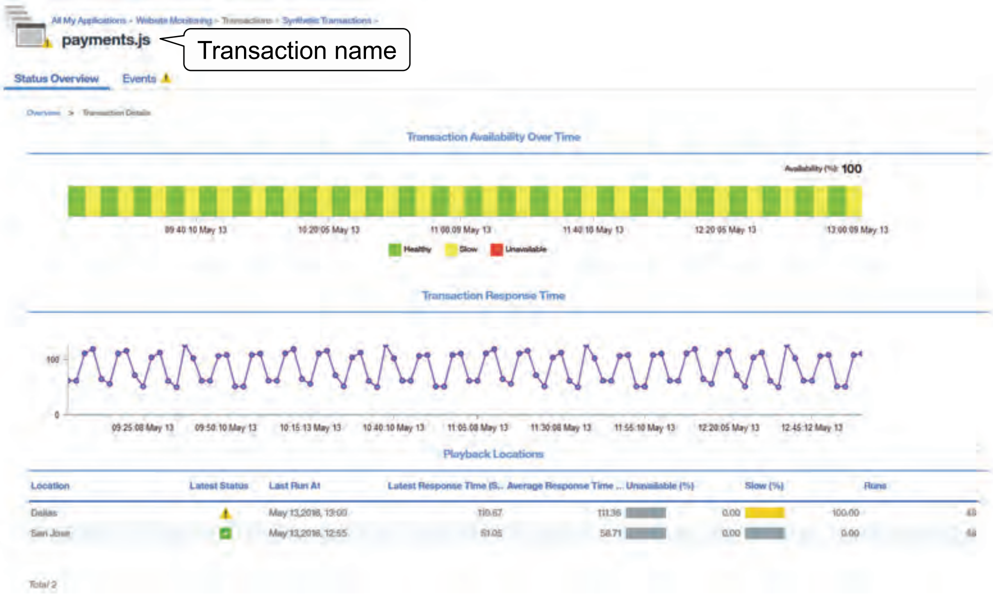
Location List

| Location | Latest Status | Last Run At | Latest Response Time (Seconds) | Average Response Time (Seconds) | Unavailable (%) | Slow (%) | Runs |
|---|---|---|---|---|---|---|---|
| Dallas | ✓ | May 13,2016, 12:10 | 5.55 | 42.53 | 0.00 | 34.00 | 140 |
| San Jose | ✓ | May 13,2016, 12:05 | 7.62 | 25.17 | 0.00 | 8.00 | 140 |

*Monitoring synthetic transactions (Cont.)*

If you are already using the Response Time agent to monitor user response time for an application, you can add a synthetic transaction to this application. You can then view more metrics and KPIs for that application in the Application Performance Dashboard.

## Monitoring synthetic transactions (Cont.)



Transaction name

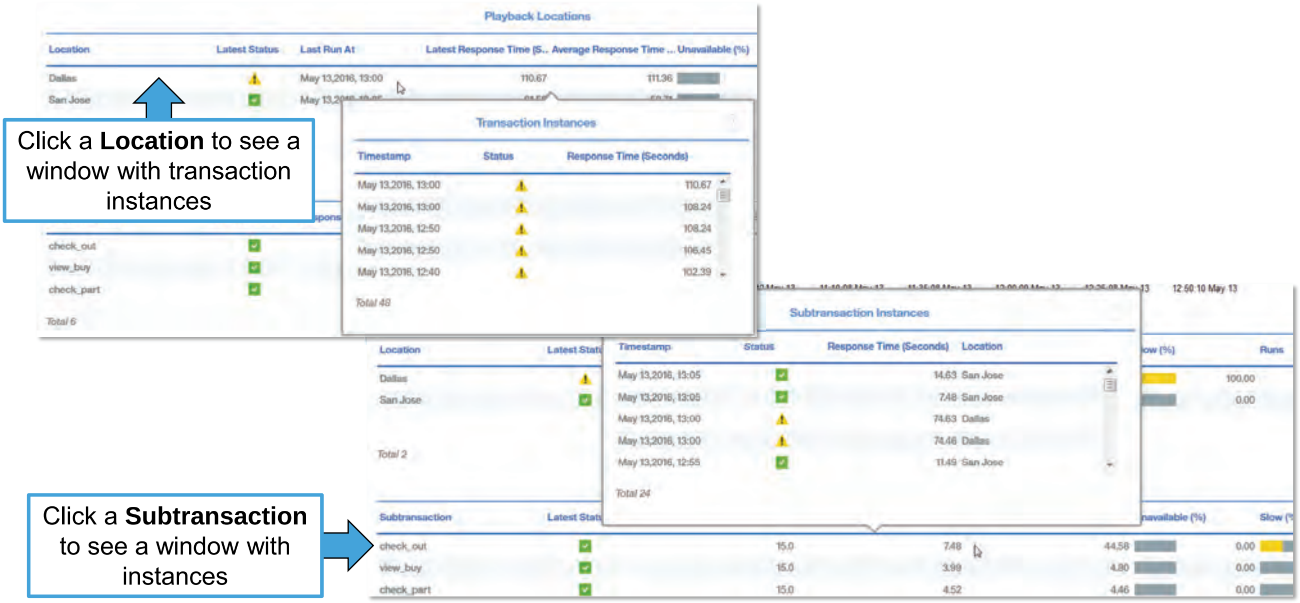
| Location | Latest Status | Last Run At | Latest Response Time (S... | Average Response Time ... | Unavailable (%) | Slow (%) | Runs |
|---|---|---|---|---|---|---|---|
| Dallas | ⚠ | May 13,2016, 13:00 | 110.67 | 111.36 | 0.00 | 100.00 | 48 |
| San Jose | ✓ | May 13,2016, 12:55 | 51.05 | 58.71 | 0.00 | 0.00 | 48 |

Total: 2

*Monitoring synthetic transactions (Cont.)*

Drilling down into a transaction (in this case, payments.js) shows performance over time and by location.

## Monitoring synthetic transactions (Cont.)



Click a **Location** to see a window with transaction instances

Click a **Subtransaction** to see a window with instances

Playback Locations

| Location | Latest Status | Last Run At | Latest Response Time (S... | Average Response Time ... | Unavailable (%) |
|---|---|---|---|---|---|
| Dallas | ⚠ | May 13,2016, 13:00 | 110.67 | 111.36 | |
| San Jose | ✓ | May 13,2016, 13:05 | | | |

| Subtransaction | Latest Status |
|---|---|
| check_out | ✓ |
| view_buy | ✓ |
| check_part | ✓ |

*Total 6*

Transaction Instances

| Timestamp | Status | Response Time (Seconds) |
|---|---|---|
| May 13,2016, 13:00 | ⚠ | 110.67 |
| May 13,2016, 13:00 | ⚠ | 108.24 |
| May 13,2016, 12:50 | ⚠ | 108.24 |
| May 13,2016, 12:50 | ⚠ | 106.45 |
| May 13,2016, 12:40 | ⚠ | 102.39 |

*Total 48*

| Location | Latest Stat | ow (%) | Runs |
|---|---|---|---|
| Dallas | ⚠ | | 100.00 |
| San Jose | ✓ | | 0.00 |

*Total 2*

Subtransaction Instances

| Timestamp | Status | Response Time (Seconds) | Location |
|---|---|---|---|
| May 13,2016, 13:05 | ✓ | 14.63 | San Jose |
| May 13,2016, 13:05 | ✓ | 7.48 | San Jose |
| May 13,2016, 13:00 | ⚠ | 74.63 | Dallas |
| May 13,2016, 13:00 | ⚠ | 74.46 | Dallas |
| May 13,2016, 12:55 | ✓ | 11.49 | San Jose |

*Total 24*

| Subtransaction | Latest Stat | | | navailable (%) | Slow (% |
|---|---|---|---|---|---|
| check_out | ✓ | 15.0 | 7.48 | 44.58 | 0.00 |
| view_buy | ✓ | 15.0 | 3.99 | 4.80 | 0.00 |
| check_part | ✓ | 15.0 | 4.52 | 4.46 | 0.00 |

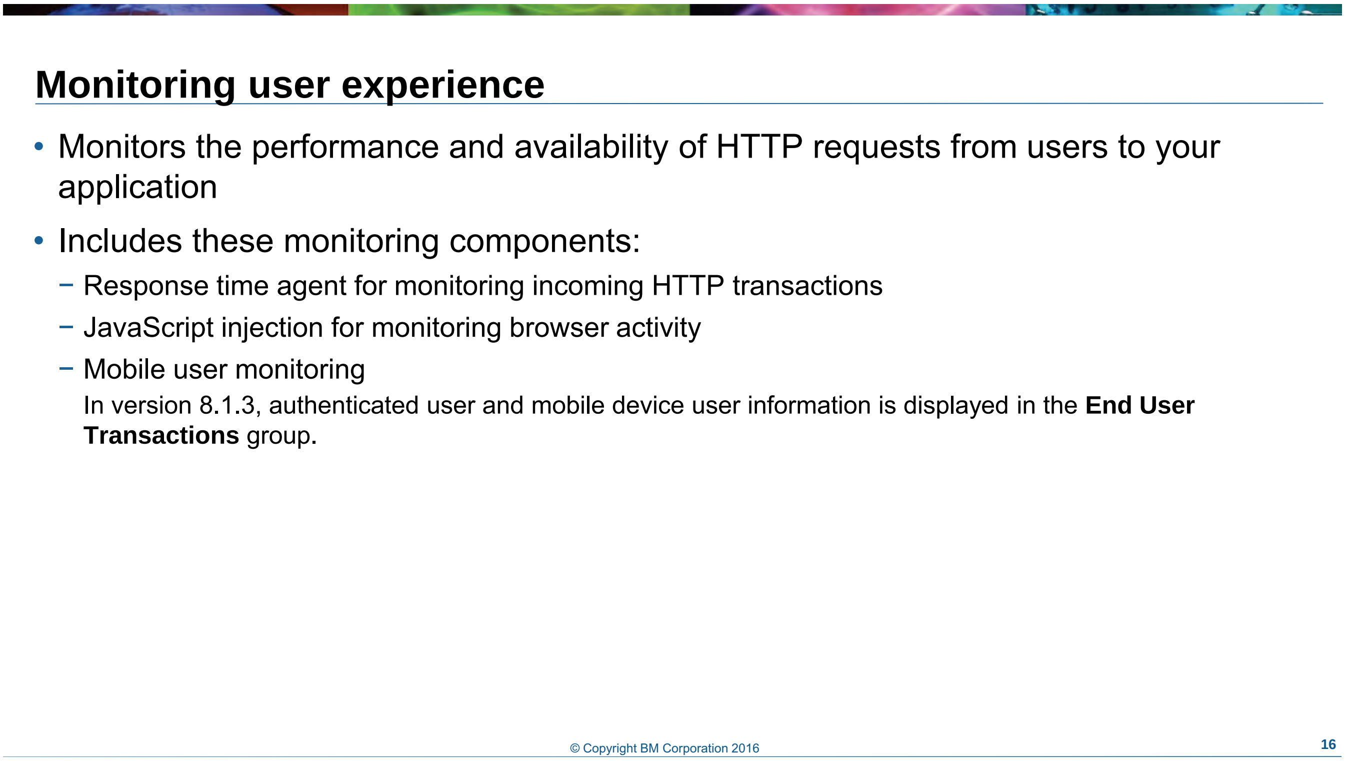
*Monitoring synthetic transactions (Cont.)*

The lowest level of drill-down in Website Monitoring takes administrators to transaction instances by location and subtransaction instances.

# Lesson 2  Monitoring users

## Lesson 2 Monitoring users

In this last section, you get an overview of how IBM Performance Management Advanced monitors users.



## Monitoring user experience

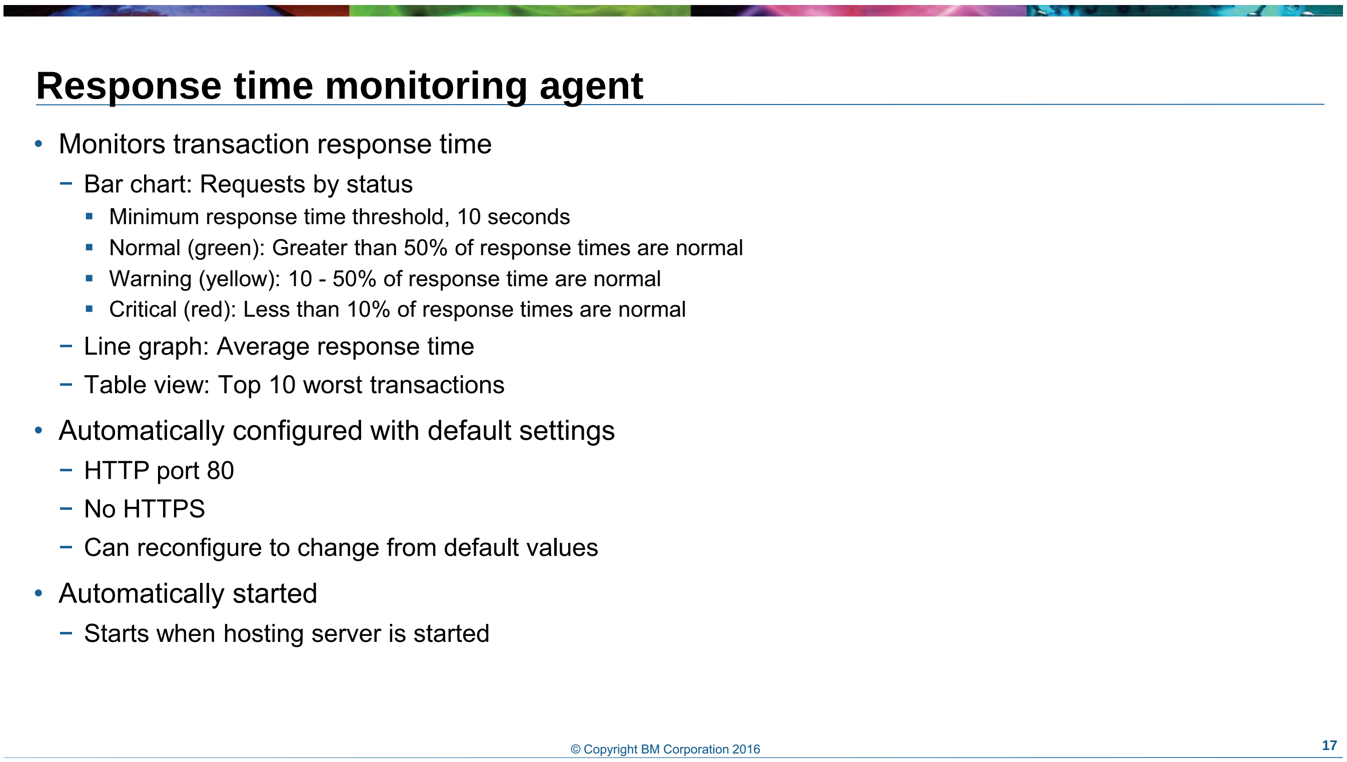
- Monitors the performance and availability of HTTP requests from users to your application
- Includes these monitoring components:
  - Response time agent for monitoring incoming HTTP transactions
  - JavaScript injection for monitoring browser activity
  - Mobile user monitoring

    In version 8.1.3, authenticated user and mobile device user information is displayed in the **End User Transactions** group.

*Monitoring user experience*

End User Transactions dashboards now include user and device information, which was previously displayed in the Authenticated Users and Mobile Devices Users dashboards in the Users group.

User, session, and device information is now sorted by location (country, state, and city) based on the IP address of the user. The updated dashboards help administrators to understand user volumes and whether issues are isolated to specific sets of users.



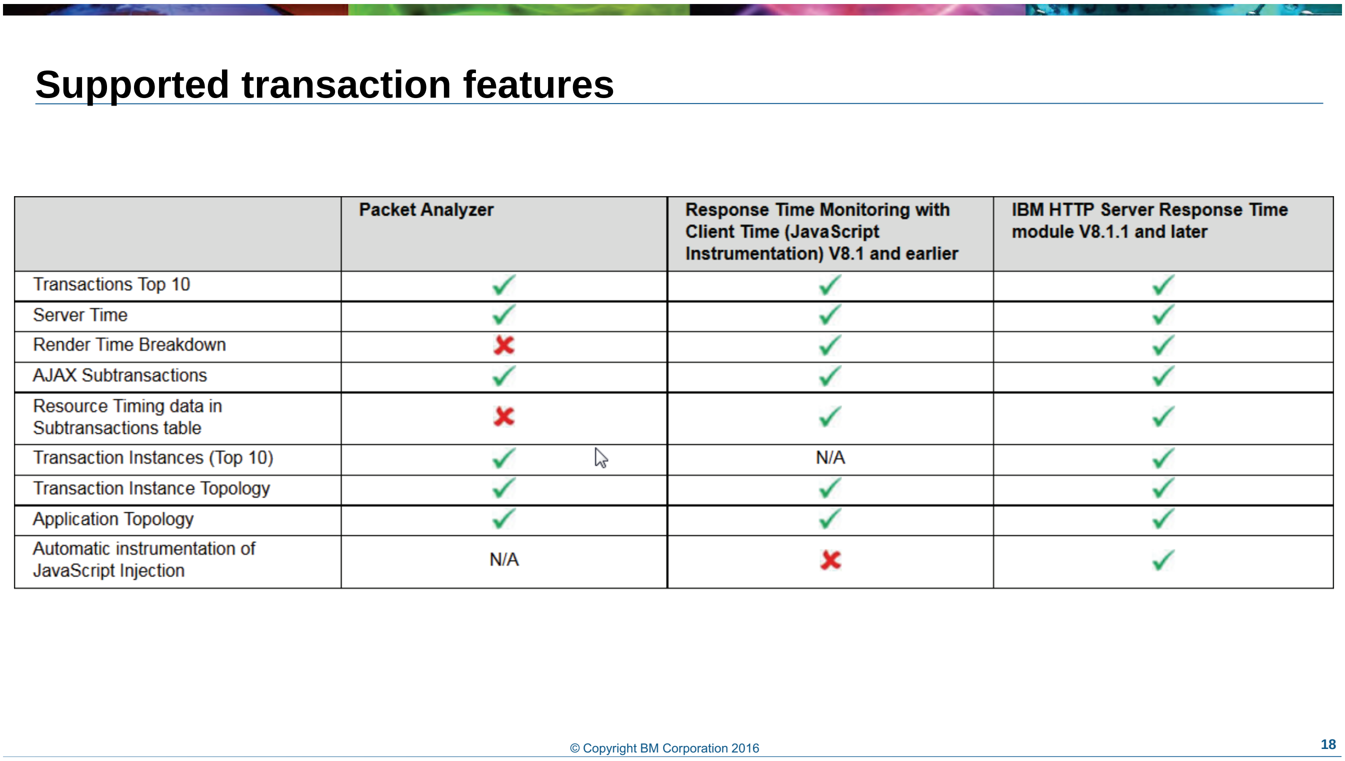
## Response time monitoring agent

- Monitors transaction response time
  - Bar chart: Requests by status
    - Minimum response time threshold, 10 seconds
    - Normal (green): Greater than 50% of response times are normal
    - Warning (yellow): 10 - 50% of response time are normal
    - Critical (red): Less than 10% of response times are normal
  - Line graph: Average response time
  - Table view: Top 10 worst transactions
- Automatically configured with default settings
  - HTTP port 80
  - No HTTPS
  - Can reconfigure to change from default values
- Automatically started
  - Starts when hosting server is started

*Response time monitoring agent*

The response time monitoring agent has these characteristics:

- The bar chart and line chart show status aggregated over all of the application’s transactions over the past two hours.
- The Top 10 shows metrics on individual transactions.
- The response time agent is a single instance. It is configured with default settings and starts automatically.



## Supported transaction features

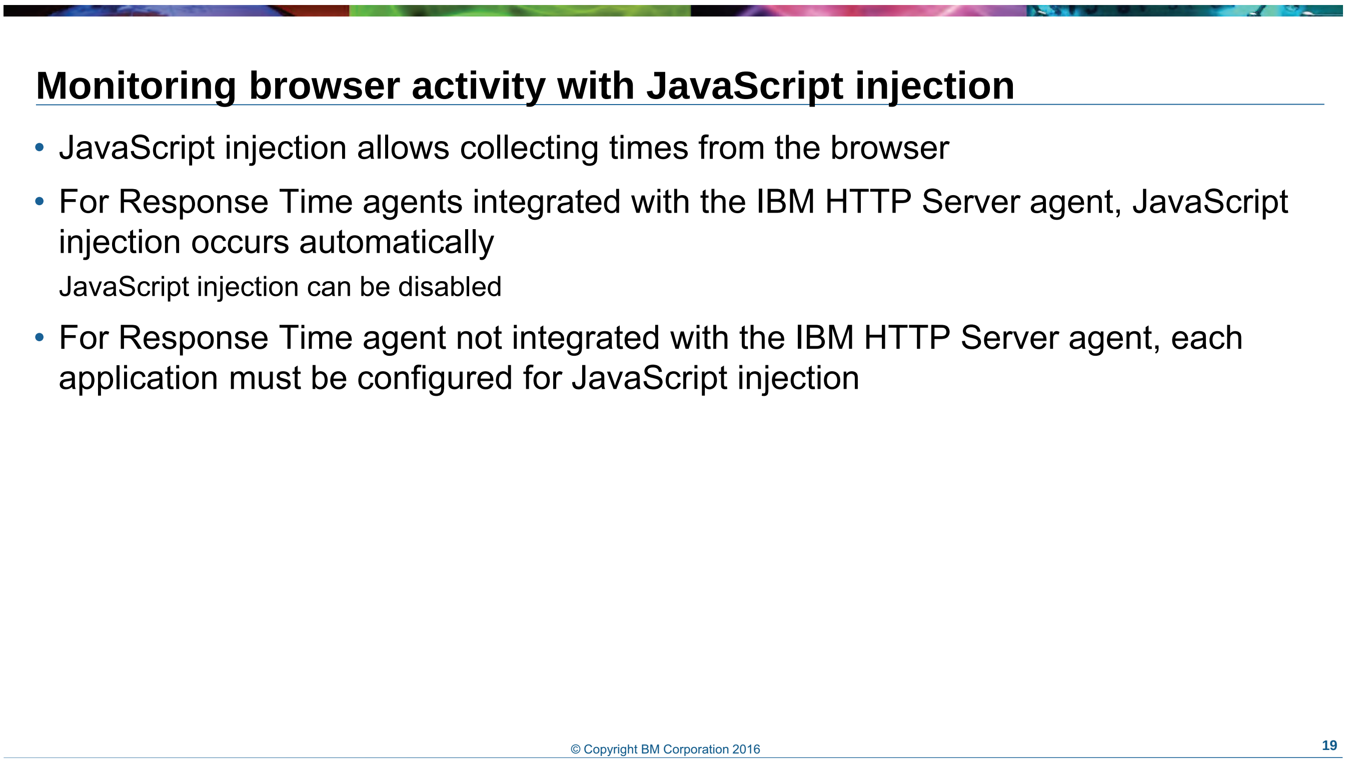
| | Packet Analyzer | Response Time Monitoring with Client Time (JavaScript Instrumentation) V8.1 and earlier | IBM HTTP Server Response Time module V8.1.1 and later |
|---|---|---|---|
| Transactions Top 10 | ✓ | ✓ | ✓ |
| Server Time | ✓ | ✓ | ✓ |
| Render Time Breakdown | ✗ | ✓ | ✓ |
| AJAX Subtransactions | ✓ | ✓ | ✓ |
| Resource Timing data in Subtransactions table | ✗ | ✓ | ✓ |
| Transaction Instances (Top 10) | ✓ | N/A | ✓ |
| Transaction Instance Topology | ✓ | ✓ | ✓ |
| Application Topology | ✓ | ✓ | ✓ |
| Automatic instrumentation of JavaScript Injection | N/A | ✗ | ✓ |

© Copyright BM Corporation 2016 18

*Supported transaction features*

IBM HTTP Server Response Time module monitors the web server port used by IBM HTTP Server. Use either the IBM HTTP Server Response Time module or Packet Analyzer to monitor IBM HTTP Server, not both.

The slide table shows the features available with the different monitors.



## Monitoring browser activity with JavaScript injection

- JavaScript injection allows collecting times from the browser
- For Response Time agents integrated with the IBM HTTP Server agent, JavaScript injection occurs automatically
  - JavaScript injection can be disabled
- For Response Time agent not integrated with the IBM HTTP Server agent, each application must be configured for JavaScript injection

© Copyright BM Corporation 2016 19

*Monitoring browser activity with JavaScript injection*

To help you understand the performance of your web pages in a browser and any errors, the Response Time Monitoring agent must be able to collect timing data from the browser. With some simple configuration to the application you want to monitor, monitoring features can collect timing data. This feature is available only in IBM Application Performance Management.

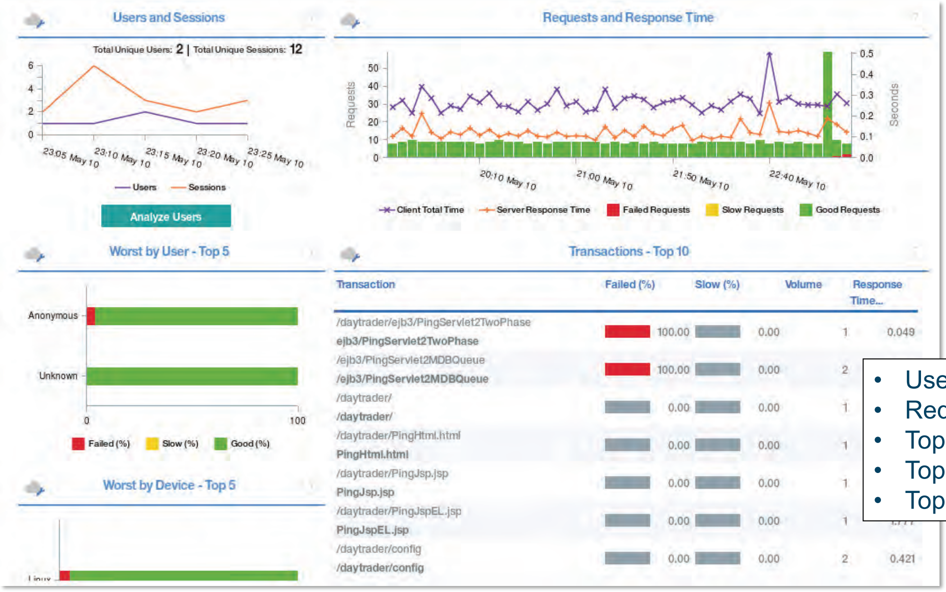
Using JavaScript, IBM HTTP Server Response Time module inserts a header into web pages that are served by an IBM HTTP Server so that Response Time Monitoring can monitor those pages. To display client-side and network data, monitored web browsers must support JavaScript and W3C performance and navigation timing. Embedded objects that are loaded by the page are tracked by using cookies. Transaction information from web pages that are served by IBM HTTP Server is then included in End User Transactions dashboards.

## *Limitations*

JavaScript code to collect browser timings is inserted automatically into a web page only if the page meets the W3C HTML standards. The following conditions must be satisfied:

- Response headers contain Content-Type: text/html
- Response content includes the <head> element

# Viewing user transactions



| Transaction | Failed (%) | Slow (%) | Volume | Response Time... |
|---|---|---|---|---|
| /daytrader/ejb3/PingServlet2TwoPhase<br>**ejb3/PingServlet2TwoPhase** | 100.00 | 0.00 | 1 | 0.049 |
| /ejb3/PingServlet2MDBQueue<br>**/ejb3/PingServlet2MDBQueue** | 100.00 | 0.00 | 2 | |
| /daytrader/<br>**/daytrader/** | 0.00 | 0.00 | 1 | |
| /daytrader/PingHtml.html<br>**PingHtml.html** | 0.00 | 0.00 | 1 | |
| /daytrader/PingJsp.jsp<br>**PingJsp.jsp** | 0.00 | 0.00 | 1 | |
| /daytrader/PingJspEL.jsp<br>**PingJspEL.jsp** | 0.00 | 0.00 | 1 | [illegible] |
| /daytrader/config<br>**/daytrader/config** | 0.00 | 0.00 | 2 | 0.421 |

- Users and Sessions
- Requests by status
- Top 10 worst transactions
- Top 5 worst user performance
- Top 5 worst device performance

*Viewing user transactions*

The request by status window shows the number of *good*, *slow*, and *failed* requests per second in 5-minute intervals, aggregated for all transactions. The chart shows the last 2 hours.

Good requests finish successfully within the minimum response time threshold, which, by default, is 10 seconds. Slow requests finish successfully, but they are above the 10-second threshold. Failed requests either do not finish at all during the sampling interval or report an error.

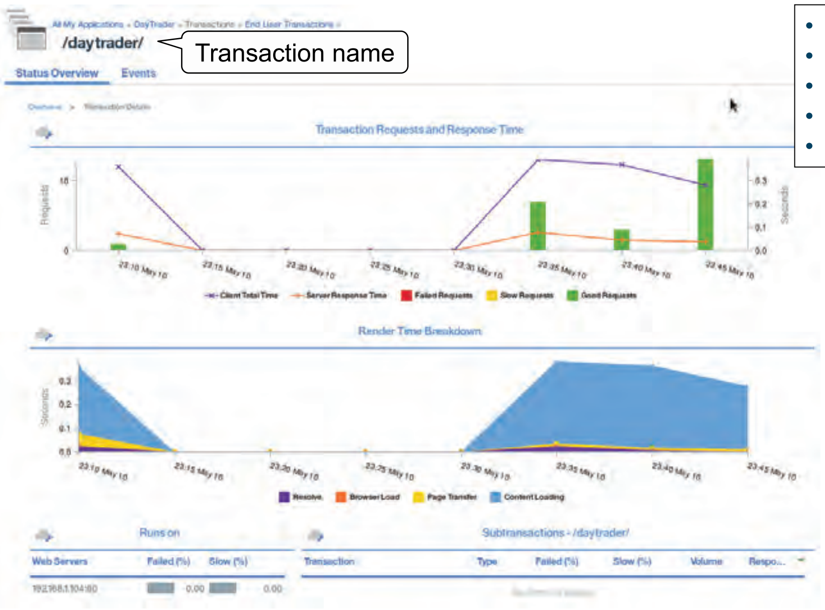
The average response time plot chart shows the average time in seconds for all the transactions. The chart also shows the last two hours.

The Transactions Top 10 view shows the 10 worst performing transactions over the sampling interval.

The response time attribute to the far right shows the average response time by transaction in the sampling interval.

You can click the rows in the transactions table to drill down into individual transactions.

## Viewing individual transaction details (APM, AD)



| Web Servers | Failed (%) | Slow (%) |
|---|---|---|
| 192.168.1.104:80 | 0.00 | 0.00 |

| Transaction | Type | Failed (%) | Slow (%) | Volume | Respo... |
|---|---|---|---|---|---|

Transaction name

- Requests by status
- Average response time
- Render Time Breakdown
- System where transaction runs
- Subtransactions (not always present)

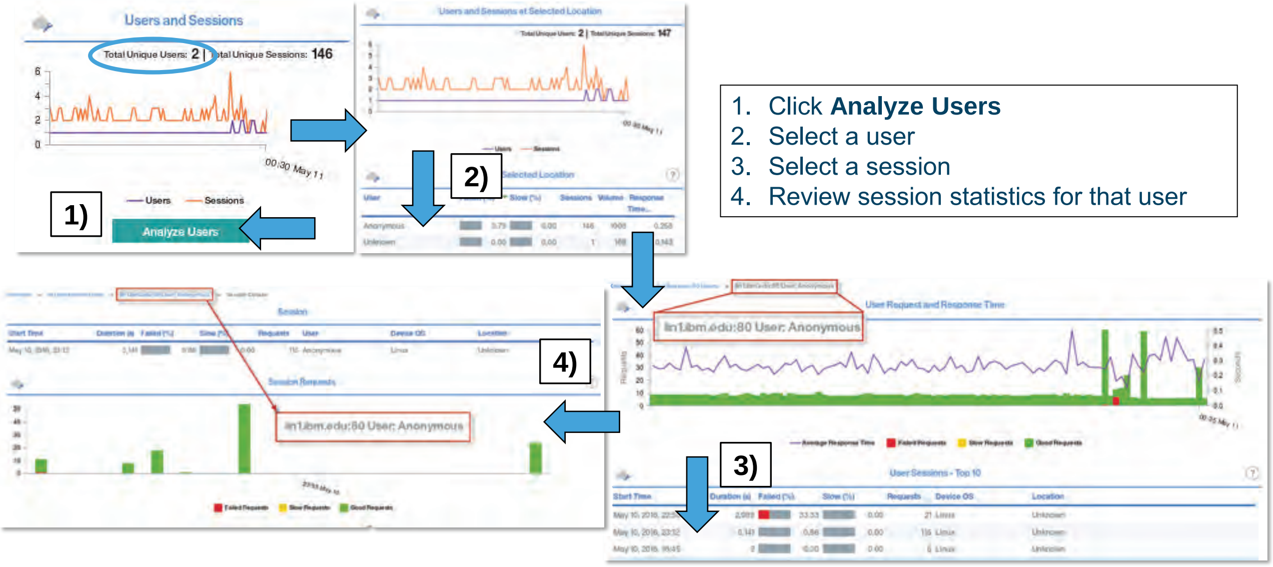
*Viewing individual transaction details (APM, AD)*

The transaction details window shows requests by status and average response time for an individual transaction. It also shows you the server or servers that the transactions run on.

Also included is a render-time breakdown of requests, including these examples:

- Resolve time
- Browser load
- Page transfer
- Content loading

## Viewing user activity


Users and Sessions
Total Unique Users: 2 | Total Unique Sessions: 146
Users
Sessions
Analyze Users
1)
Users and Sessions at Selected Location
Total Unique Users: 2 | Total Unique Sessions: 147
Selected Location
2)
lin1.ibm.edu:80 User: Anonymous
User Request and Response Time
User Sessions - Top 10
3)
Session
Session Requests
4)

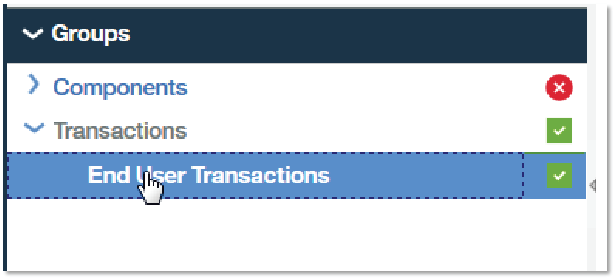
1. Click **Analyze Users**
2. Select a user
3. Select a session
4. Review session statistics for that user

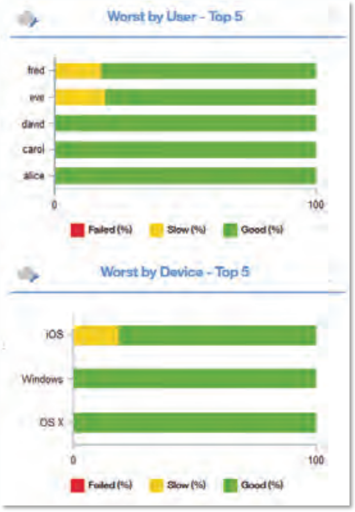
*Viewing user activity*

The Users and Sessions widget lists those statistics for the current data collection period. Individual users can drill down to the session level.

## Monitoring mobile device user transactions

- Install and configure the Response Time Monitoring agent and the Transaction Tracking agent and the associated dashboards to monitor the performance and availability of HTTP and HTTPS for Mobile devices that are accessing your environment.
- Access **user Transaction** activity in the **Groups** widget.
- Drill down by worst-performing users or by device (categorized by OS)

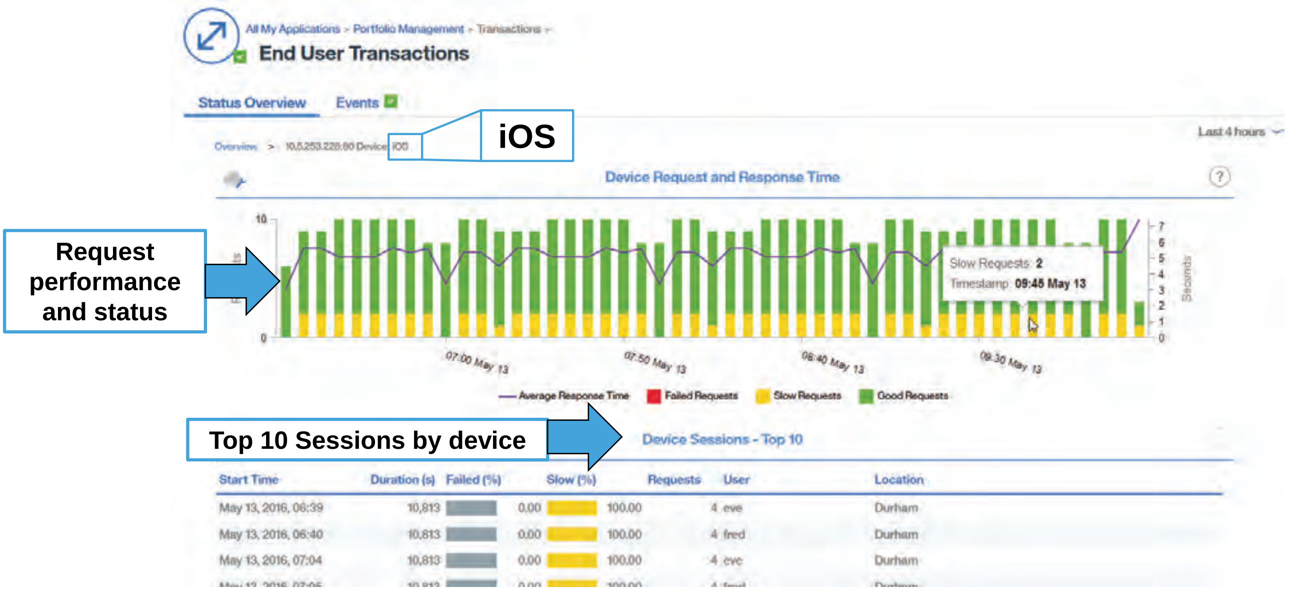




*Monitoring mobile device user transactions*

Install and configure the Response Time Monitoring agent and the Transaction Tracking agent and the associated dashboards to monitor the performance and availability of HTTP and HTTPS for mobile devices access your environment. You can track performance by device and by user.

## User activity by device (operating system)



All My Applications > Portfolio Management > Transactions >

End User Transactions

Status Overview | Events

Overview > 10.5.253.228:80 Device iOS

Last 4 hours

Device Sessions - Top 10

| Start Time | Duration (s) | Failed (%) | Slow (%) | Requests | User | Location |
|---|---|---|---|---|---|---|
| May 13, 2016, 06:39 | 10,813 | 0.00 | 100.00 | 4 | eve | Durham |
| May 13, 2016, 06:40 | 10,813 | 0.00 | 100.00 | 4 | fred | Durham |
| May 13, 2016, 07:04 | 10,813 | 0.00 | 100.00 | 4 | eve | Durham |
| May 13, 2016, 07:05 | 10,813 | 0.00 | 100.00 | 4 | fred | Durham |

*User activity by device (operating system)*

You can monitor user activity by device, that is, operating system.

Look at the Mobile Active Sessions graph to see the number of open sessions serving mobile devices on the application server over a recent period. You can use the graph to detect any peak loads.

Look at the Active Sessions by Mobile OS - Top 5 chart to compare the current number of active sessions per mobile operating system.

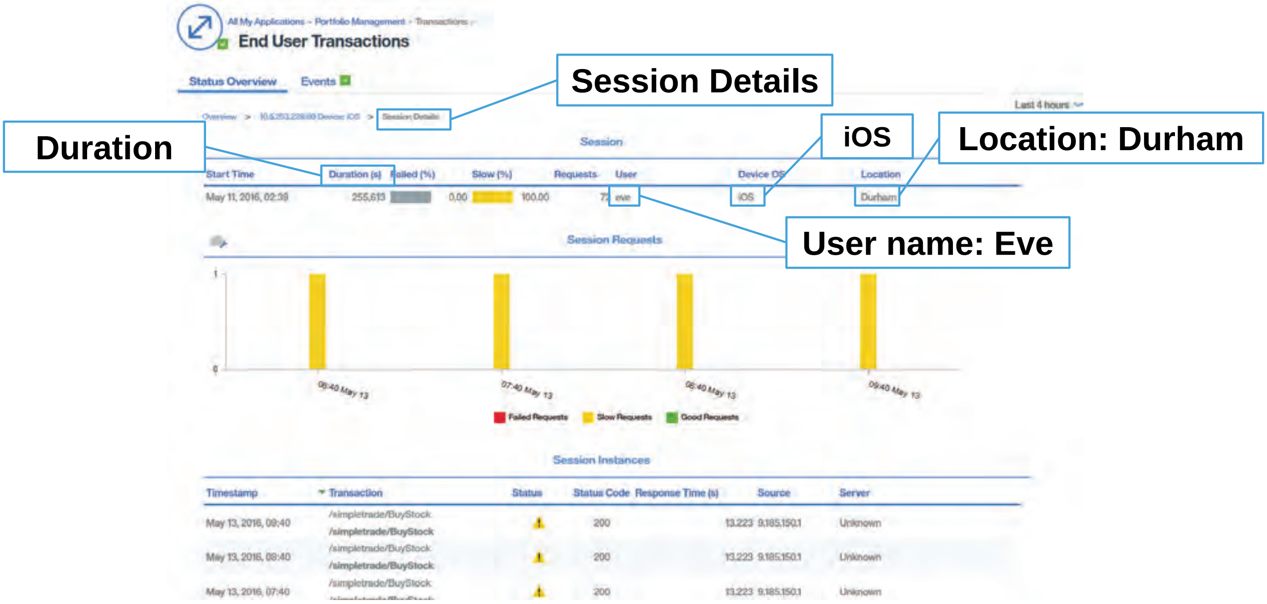
Look at the Session Duration by Mobile OS - Top 5 chart to compare the average session duration, that is, how long a session is running, per mobile operating system. This value is an average for all sessions that are currently open.

Look at the Requests by Mobile OS - Top 5 to compare the number of web requests sent by devices using various mobile operating systems over a recent period.

Look at the Percentage Failures by Mobile OS - Top 5 chart to see how many of the total number of requests failed (red), took too long to process (yellow), or were good (green) for each device running a mobile operating system.

Click an operating system to open the Mobile Devices Users Details dashboard.

## Session details by device (operating system)



All My Applications - Portfolio Management - Transactions -

End User Transactions

Status Overview Events

Overview > 10.6.253.228:80 Device: iOS > Session Details

Last 4 hours

Session

| Start Time | Duration (s) | Failed (%) | Slow (%) | Requests | User | Device OS | Location |
|---|---|---|---|---|---|---|---|
| May 11, 2016, 02:39 | 255,613 | 0.00 | 100.00 | 72 | eve | iOS | Durham |

Session Requests

Session Instances

| Timestamp | Transaction | Status | Status Code | Response Time (s) | Source | Server |
|---|---|---|---|---|---|---|
| May 13, 2016, 09:40 | /simpletrade/BuyStock<br>**/simpletrade/BuyStock** | ⚠ | 200 | 13.223 | 9.185.150.1 | Unknown |
| May 13, 2016, 08:40 | /simpletrade/BuyStock<br>**/simpletrade/BuyStock** | ⚠ | 200 | 13.223 | 9.185.150.1 | Unknown |
| May 13, 2016, 07:40 | /simpletrade/BuyStock | ⚠ | 200 | 13.223 | 9.185.150.1 | Unknown |

Session Details

Duration

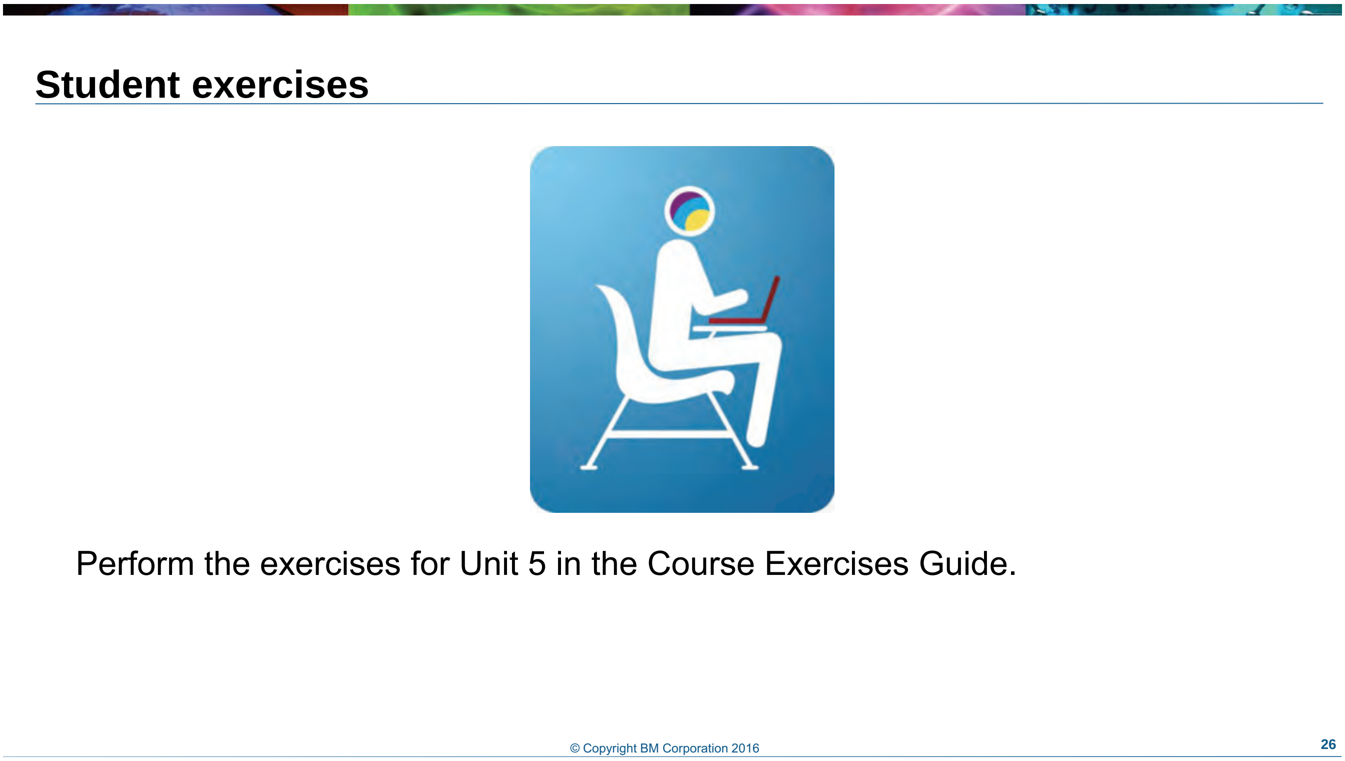
iOS

Location: Durham

User name: Eve

*Session details by device (operating system)*

At the Session Details level view, the performance of individual sessions, including the user name, device OS, duration, and location.

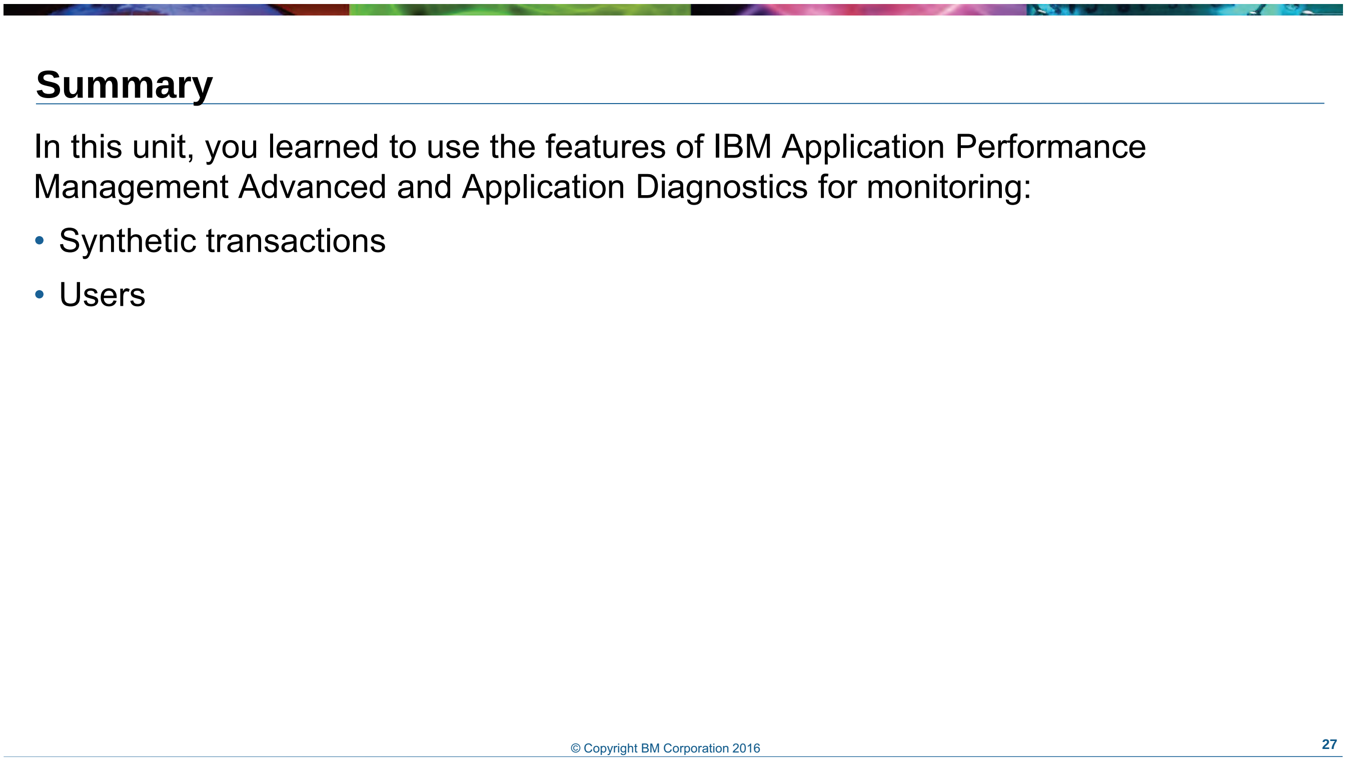


## Student exercises

Perform the exercises for Unit 5 in the Course Exercises Guide.

*Student exercises*

You now perform the exercises for Unit 5 in the Course Exercises Guide.



## Summary

In this unit, you learned to use the features of IBM Application Performance Management Advanced and Application Diagnostics for monitoring:

- Synthetic transactions
- Users

© Copyright BM Corporation 2016

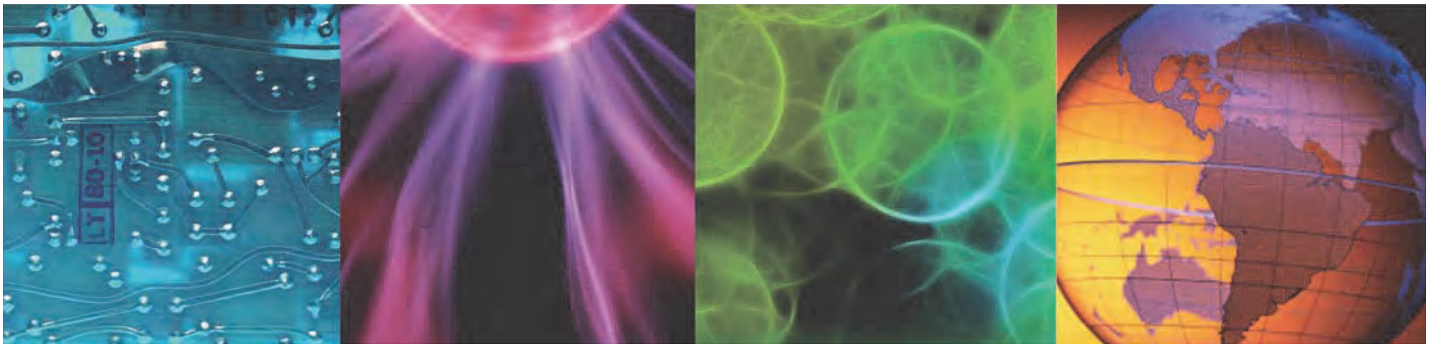
27

*Summary*

In this unit, you learned to use the features of IBM Application Performance Management Advanced and Application Diagnostics for monitoring:

- Synthetic transactions
- Users

TOD45 1.0



ibm.com/training

Authorized IBM Training