

Settings – Server-side Services

Dynatrace Training Module



Agenda

- Custom Services
- Merged Service Monitoring
- Request Attributes
- API Detection Settings
- Deep Monitoring

Custom Services

Custom Services

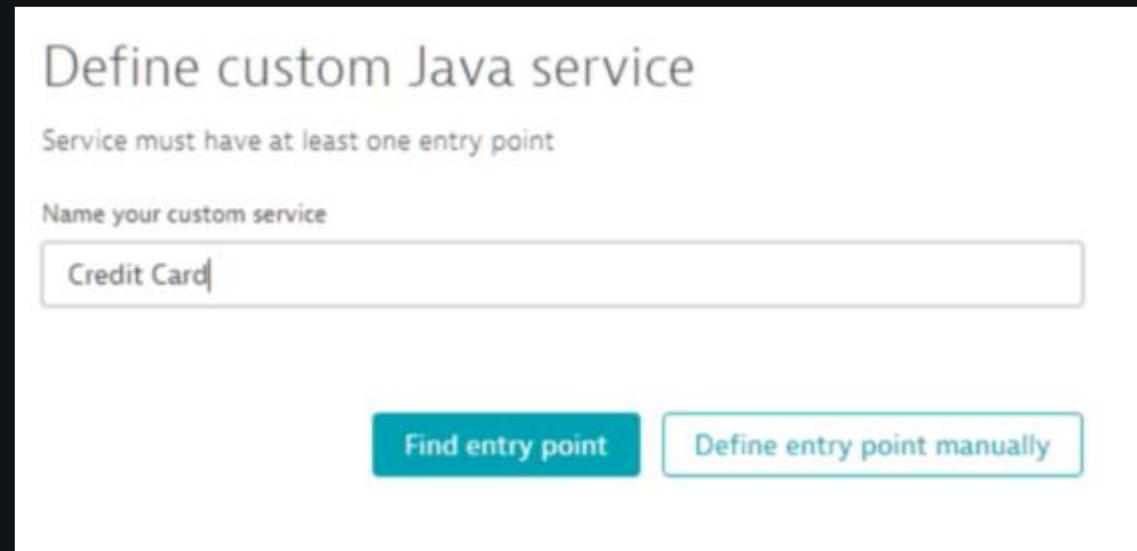
- Dynatrace automatically detects and monitors most server-side services in your environment with no configuration required
- If your application doesn't rely on standard frameworks, you can set up custom services
- With a custom service you can instruct Dynatrace which method, class, or interface it should use
- Only Java and .Net is currently supported

The screenshot shows the Dynatrace Settings interface with the following details:

- Left sidebar:** Settings > Server-side service monitoring > Custom service detection.
- Left navigation:** Monitoring, Web and mobile monitoring, Cloud and virtualization, Server-side service monitoring (selected), Custom service detection, Log analytics.
- Central content:** **Custom service detection** section. It explains that Dynatrace automatically detects and monitors most server-side services. It also mentions that with a custom service, you can instruct Dynatrace which method, class, or interface it should use to gain access to each of your application's custom server-side services.
- Java services tab:** Selected. It says "Enable this setting to have changes to your custom services applied to Java processes in real-time with no required restart." A radio button is selected next to "Enable real-time updates to Java services".
- Define Java service:** A button to "Define Java service".
- Table:** Shows columns for Active, Service name, Entry points, and Move up/down. It displays the message "No services".

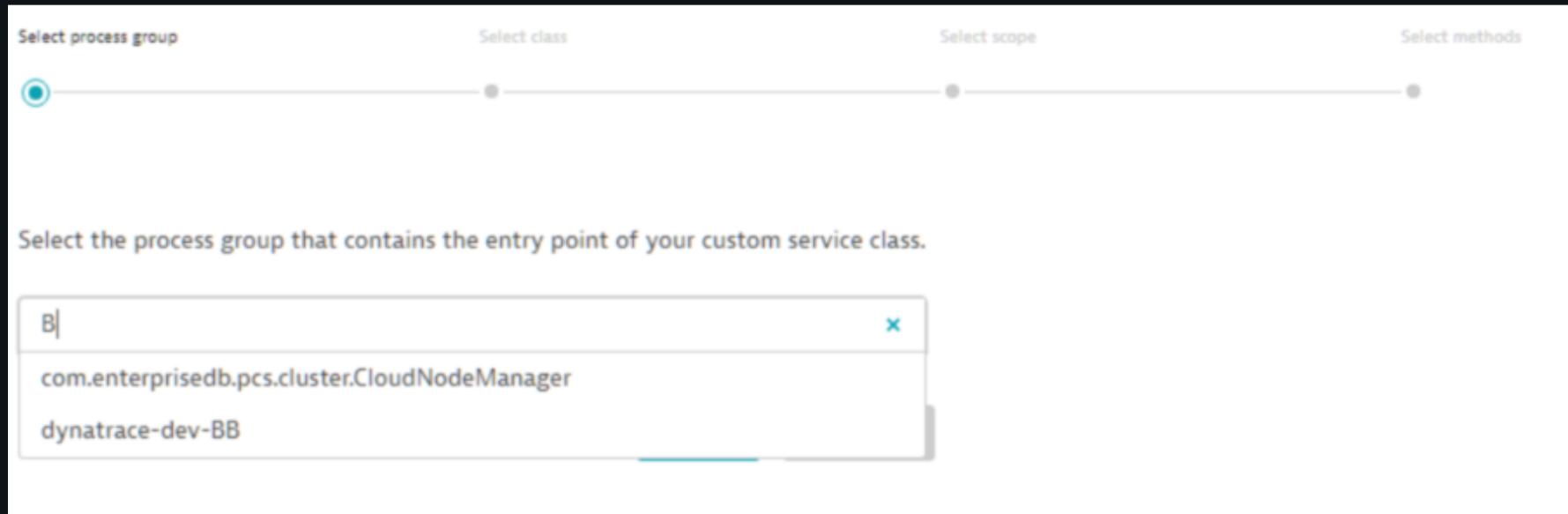
Custom Services

- Provide a Name for your custom service
- Click the Find entry point button



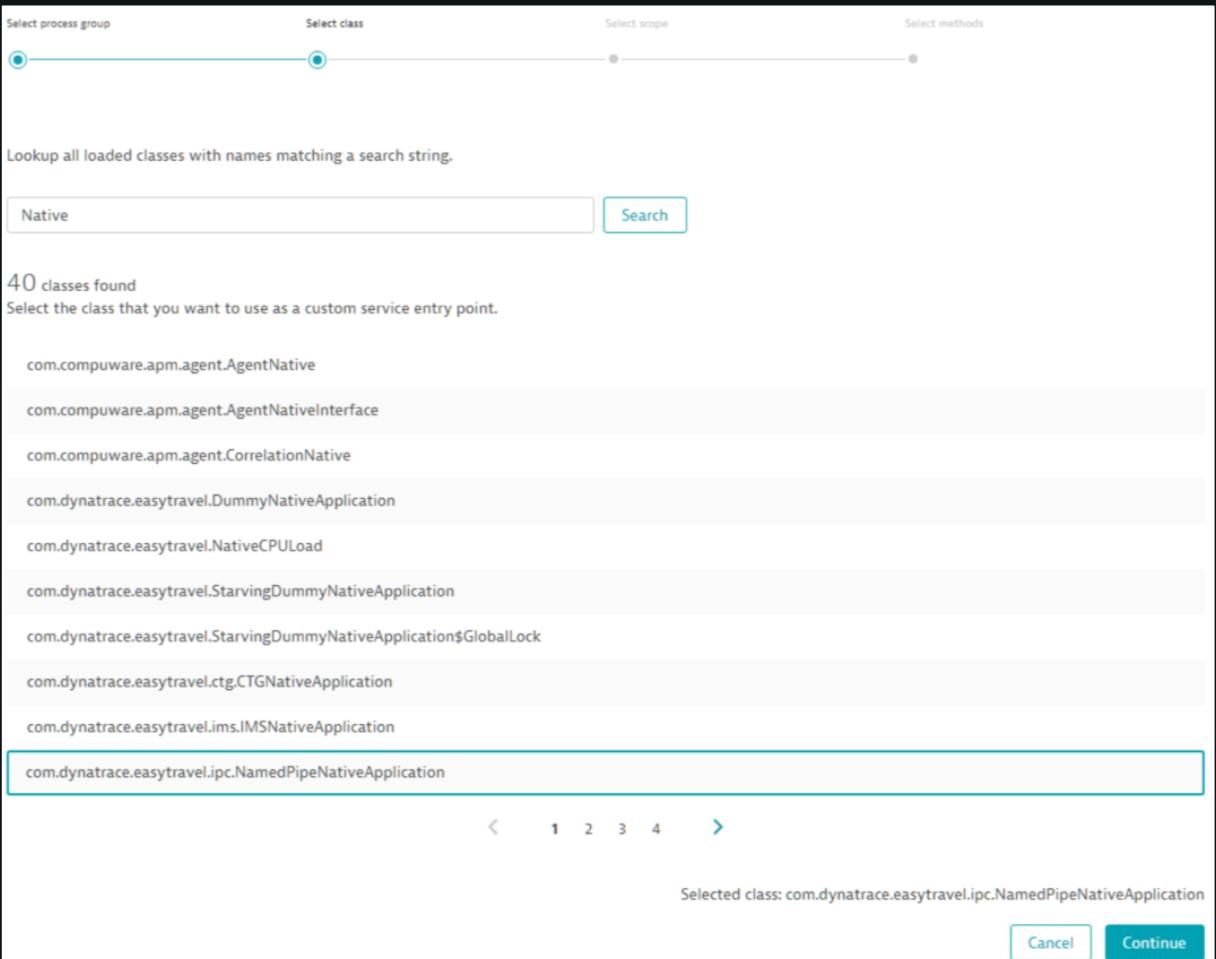
Custom Services

- Select the process group that contains the entry point of the custom service you want to monitor.
- Begin typing in the Select the process group text field to filter the process group list



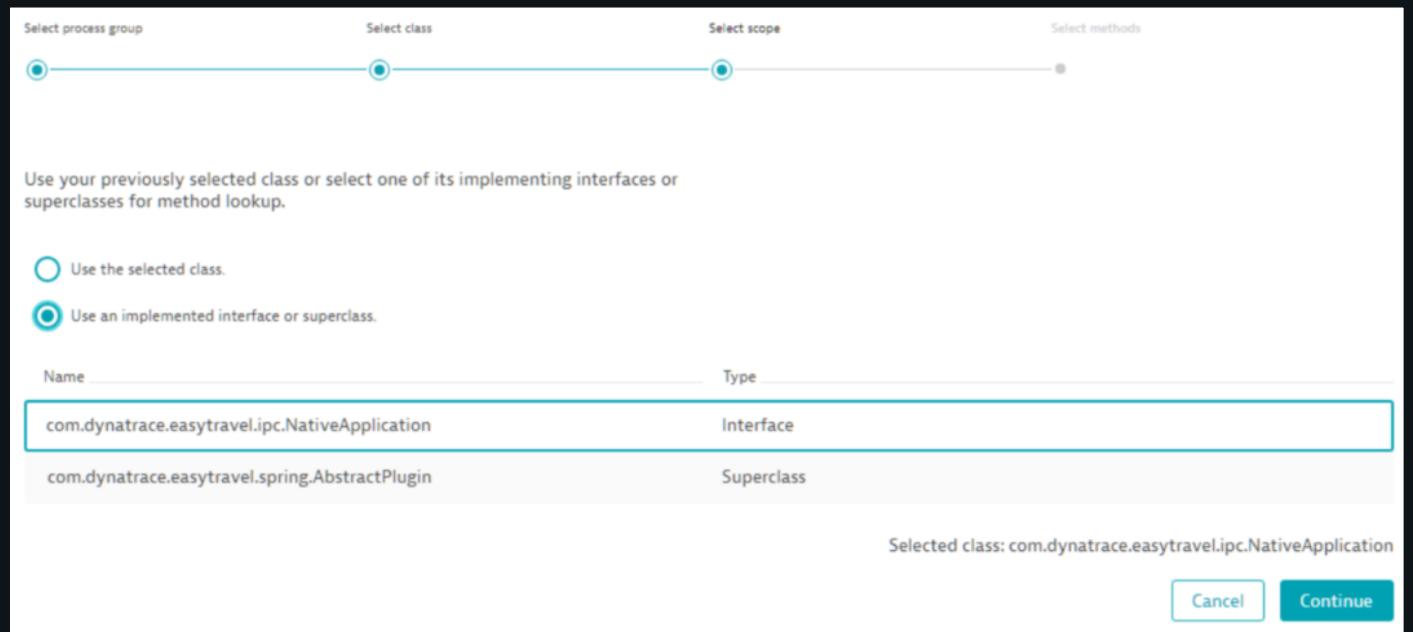
Custom Services

- Search for the class
- Type in part of the class name and click the Search button to locate the class
- Refine your search if the class you want isn't listed



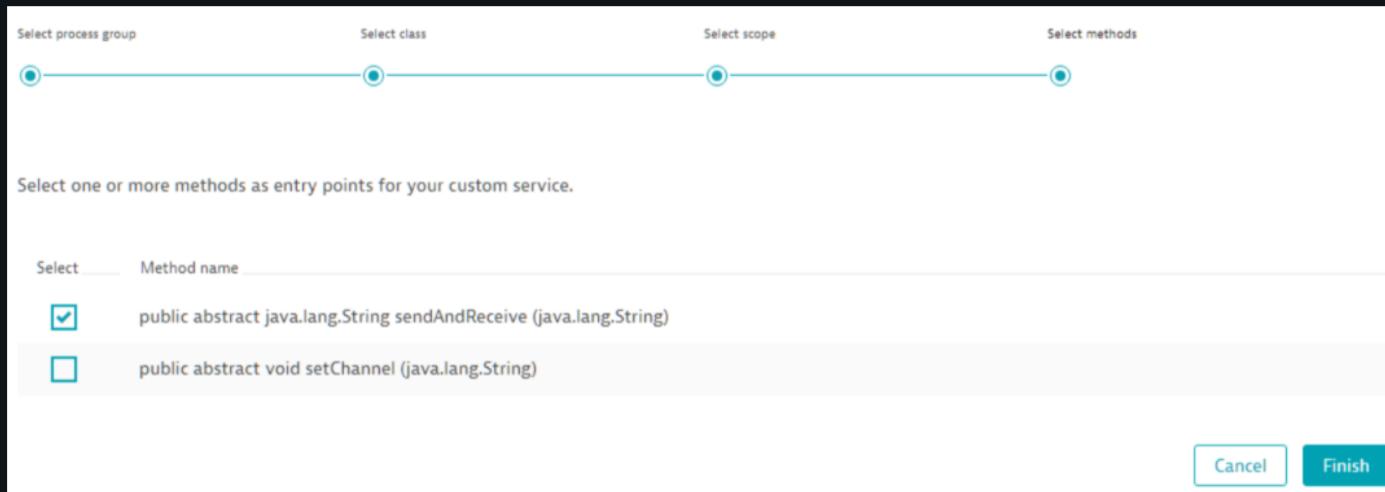
Custom Services

- (optional)
- If the selected class has super classes or an interface, select the Use an implemented interface or superclass option button to use one of these instead of the implementation you selected
- Otherwise, select Use the selected class



Custom Services

- Select the methods of the class that should act as entry points to the custom service
 - Click Finish to save the service's entry point configuration
-
- Java only
 - Set the "Enable real-time updates to Java services" switch to the On position



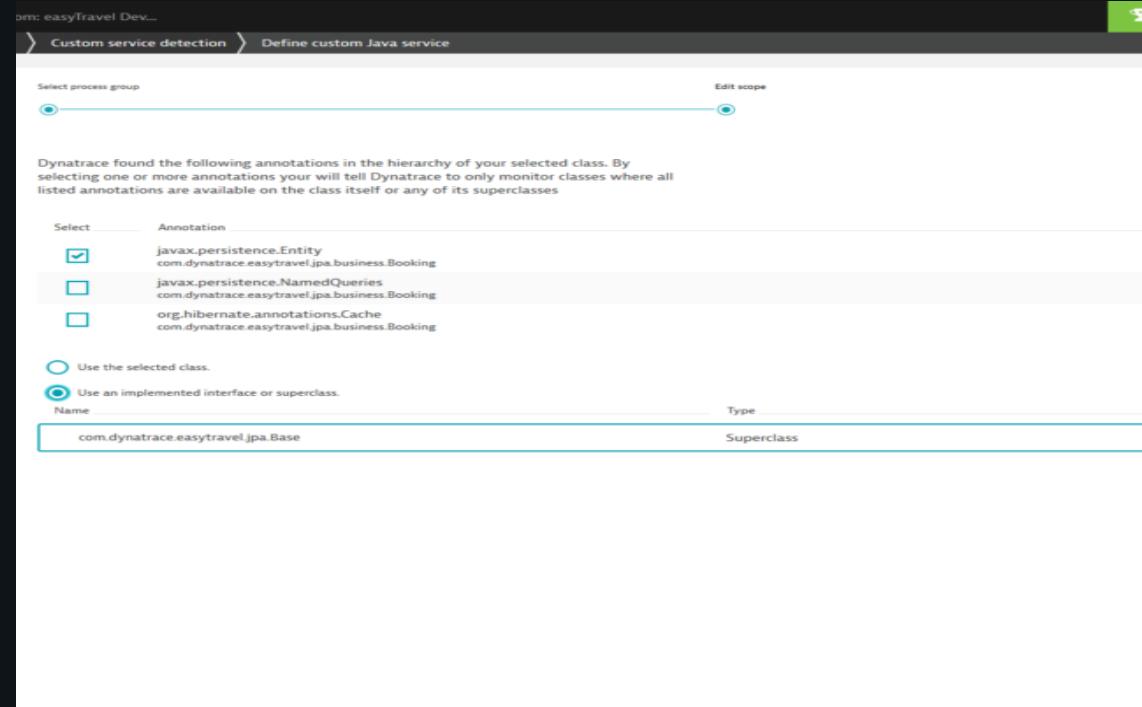
Use Java Annotations to define a custom service

- Many frameworks use interfaces but more and more rely on annotations to designate components.
- Instead of thinking about annotations right away simply define your custom service based on one particular incarnation of your component.
- After the custom service is created a new option to 'edit scope' will appear.

The screenshot shows the Dynatrace interface for defining a custom Java service. The left sidebar navigation includes 'Monitoring', 'Process groups', 'Web and mobile monitoring', 'Cloud and virtualization', 'Server-side service monitoring' (selected), 'Custom service detection', 'Log Analytics', 'Anomaly detection', 'Alerting', and 'Integration'. The main content area is titled 'Define custom Java service' with a sub-section 'Name your custom service' containing the text 'Test Annotation'. Below this is a 'Entry points' section with a note about entry points being methods or implementations of an interface. It includes a checkbox for 'This service is used inside a busy loop to process dequeued messages of a queue' and a note about JMS MessageListener and RabbitMQ DefaultConsumer. There are buttons for 'Find another entry point' and 'Define entry point manually'. A table lists an active entry point: 'com.dynatrace.easytravel.jpa.business.Booking'. To the right, there are sections for 'Fully qualified class name' (set to 'com.dynatrace.easytravel.jpa.business.Booking'), 'Methods' (set to 'public java.lang.String getId()'), and 'Annotation filter' (with a note that no annotations are selected). Buttons for 'Save' and 'Cancel' are at the top right.

Java Annotations

- Next, choose the process you want to ask for class meta data.
- Dynatrace will then show you all the annotation it found on the class.
- Most importantly this feature looks for all specific annotations in all super classes. This enables indirect matching scenarios where the annotation is on a base class not directly related to the interface match.



Merged Service Monitoring

Merged Service Monitoring

- Merged service monitoring enables you to set up monitoring rules that instruct Dynatrace to consider separately detected services as being logically the same service
- This is useful in cases where you have multiple virtual hosts, context roots, or listen ports that represent the same logical entity
- This approach simplifies monitoring and has no effect on the underlying services themselves

The screenshot shows the Dynatrace Settings interface with the following navigation path: Settings > Service detection > Merged service monitoring.

The left sidebar lists several sections:

- Monitoring: Setup and overview, Monitoring overview, Process group detection, Monitored technologies
- RUM/Mobile monitoring: Global settings and configuration
- Cloud and virtualization: Connect vCenter, AWS, or OpenStack
- Service detection: Manage & customize service detection (this section is expanded, showing "Merged service monitoring" as a sub-item)
- Custom services
- Log analytics: Customize detection of log-based events

The main content area is titled "Merged service monitoring" and contains the following text:
Merged service monitoring enables you to set up monitoring rules that instruct Dynatrace to consider separately detected services as being logically the same service. This can be achieved by merging context roots, or listen ports that represent the same logical entity. This approach simplifies monitoring and has no effect on the underlying services themselves.

A teal button labeled "+ Create merged service" is visible. Below it, a message states "22 mergeable services found." A list of merged services is shown, each with a thumbnail icon and the name: "nginxForMicroservices" and "nginxForCustomerFrontend".

Merged Service Monitoring

- Say for example that you have an Apache web server with multiple virtual host definitions for *dynatrace.com*, *dynatrace.de*, and *dynatrace.at*
- From the perspectives of Apache and Dynatrace these are independent virtual hosts and services
- For your monitoring purposes however, you might want to view these services as a single logical service called *Dynatrace web page*

Merged Service Monitoring

The screenshot shows a navigation path at the top: Settings > Service detection > Merged service monitoring > Mergeable services. The main area is titled "Mergeable services" with the sub-instruction "Select any service from the list." Below is a search bar and a list of services:

Service	Description
pl11-comp02.maas:433	Apache Web Server apache2 (No requests for 21 h 40 min)
easyTravelAdmin (/easyTravelMonitor)	eT-demo-2-BusinessBackend (eT-demo-2-BusinessBackend-2-WIN1)
EasyTravelWebserver:8999	eT-demo-2-Frontend-LoadBalancer
Request on port 8091	Detecting service... (No requests for 4 h 21 min)
Request on port 9000	Detecting service...
weather-service-restify	weather-service-restify
pl11-comp02.maas:35347	Apache Web Server apache2

Merged Service Monitoring

The screenshot shows the 'Create merged service' page within the 'Settings' menu. The left sidebar contains a navigation tree with the following structure:

- Monitoring
 - Setup and overview
- RUM/Mobile monitoring
 - Global settings and configuration
- Cloud and virtualization
 - Connect vCenter, AWS, or OpenStack
- Service detection
 - Manage & customize service detection
 - Custom services
 - Merged service monitoring**
- Log analytics
 - Customize detection of log-based events
- Anomaly detection
 - Configure detection sensitivity

The 'Merged service monitoring' option is highlighted in blue. The main content area is titled 'Create merged service' and displays the following instructions: 'Add at least two services from the Mergeable services list.' Below this, there is a text input field labeled 'Name of merged service' with a red asterisk indicating it is required, followed by a 'Create' button and a 'Cancel' button. The 'Mergeable services' section lists four items:

- pl11-comp02.maas:433
Apache Web Server apache2
(No requests for 21 h 40 min)
- pl11-comp02.maas:35347
Apache Web Server apache2
- pl11-comp02.maas:70
Apache Web Server apache2
(No requests for 10 h 42 min)
- 172.17.0.2:80
Apache Web Server apache2
- pl11-comp02.maas:4990
Apache Web Server apache2

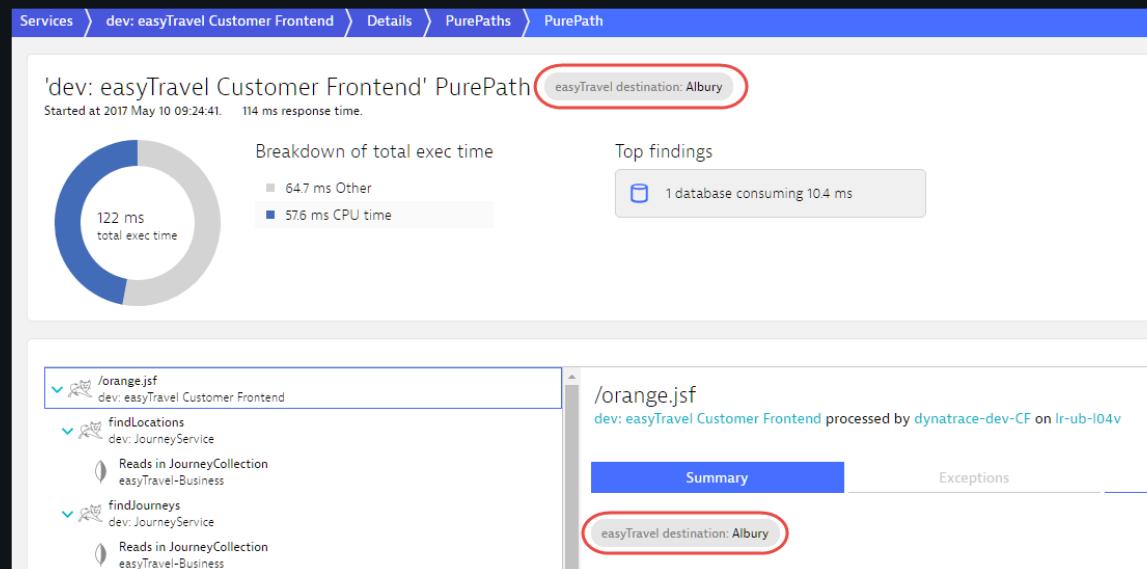
Request Attributes

Request Attributes

- Dynatrace tracks all requests from end-to-end and automatically monitors the services that underlie each transaction
- The performance and attributes of each request can be analyzed in detail
- You can even create custom, multi-faceted filters that enable you to analyze call sequences from multiple angles
- With such advanced request filtering, Dynatrace enables you to slice and dice your way through your requests to find the proverbial “needle in a haystack.”
- Until now such filtering was only possible on certain predefined attributes
- You can now configure custom request attributes that you can use to improve filtering and analysis of problematic web requests

Request Attributes

- Request attributes are essentially key/value pairs that are associated with a particular service request
- For example, if you have a travel website that tracks the destinations of each of your customers' bookings, you can set up a destination attribute for each service request
- The specific value of the destination attribute of each request is populated for you automatically on all calls that include a destination attribute



Create a Request Attribute

- Go to Settings
- Click the Create new request attribute button
- Provide a unique Request attribute name
- Request attributes can have one or more rules
 - Rules define how attribute values are fetched

The screenshot shows the Dynatrace Settings interface. The left sidebar has a tree structure with nodes like Monitoring, Web and mobile monitoring, Cloud and virtualization, Server-side service monitoring (which is expanded), and Request attributes. The main content area is titled "Rule based request attributes" and contains a description of what request attributes are and how they can be used for filtering and searching. Below this is a table titled "Create new request attribute" with four rows: "X-Dynatrace", "X-Global-Transaction-ID", "X-correlationId", and "destination". Each row has "Edit" and "Enable" buttons at the top right.

Rule	Edit	Enable
X-Dynatrace	<input type="button" value="Edit"/>	<input checked="" type="button" value="Enable"/>
X-Global-Transaction-ID	<input type="button" value="Edit"/>	<input checked="" type="button" value="Enable"/>
X-correlationId	<input type="button" value="Edit"/>	<input checked="" type="button" value="Enable"/>
destination	<input type="button" value="Edit"/>	<input checked="" type="button" value="Enable"/>

Create a Request Attribute

- Data type
 - Text, Integer, Double
- Aggregation
 - First/Last occurrence
 - Set of distinct values
 - Min/Max/Avg/Sum of captured values
 - Count occurrences/distinct
- Normalize text
 - Original/to upper/to lower

Request attributes

Dynatrace can define request attributes based on captured data from requests to facilitate advanced filtering. Based on custom data sources that you define, request attributes can serve as key/value attributes that are filterable across all Dynatrace service and PurePath views. Request attribute values can be extracted from Web request URLs, HTTP request headers, or other request metadata.

Request attribute name

Name your request attribute

Data type (immutable after setup)

Text

Aggregation on request

First occurrence

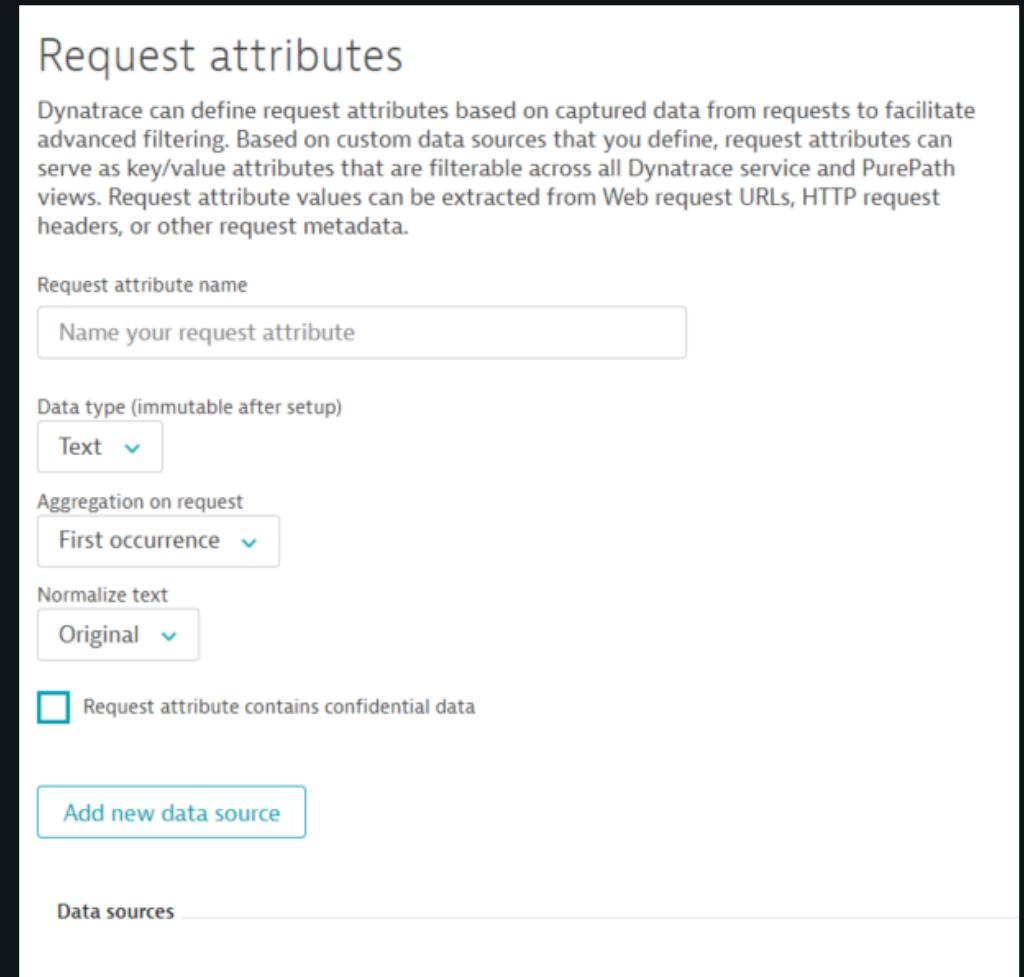
Normalize text

Original

Request attribute contains confidential data

Add new data source

Data sources



Create a Request Attribute

- Data Sources
 - HTTP Request/Response header
 - Web request URL
 - Web request URL query parameter
 - HTTP POST parameter
 - Java method
 - .Net method

Rule applies to requests of the following process group, technology and with process group tag

All process groups ▾

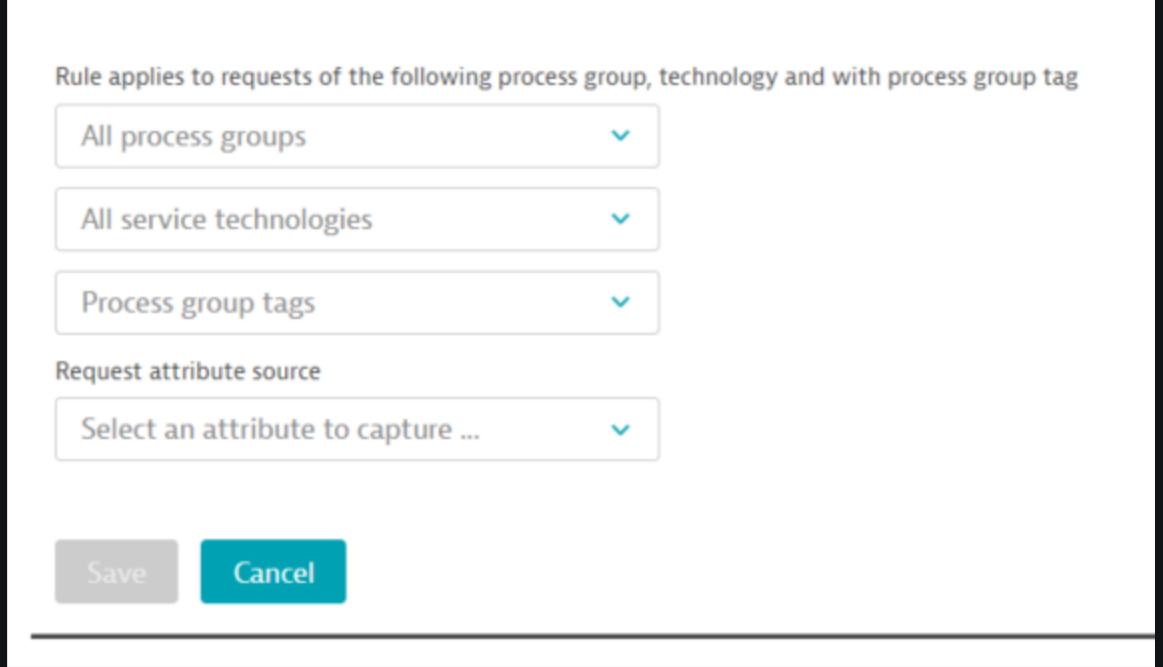
All service technologies ▾

Process group tags ▾

Request attribute source

Select an attribute to capture ... ▾

Save **Cancel**



Capturing Method Arguments

- Select the process group that contains the classes or interfaces you're interested in
- Search for the class that includes the method you're interested in

Lookup all loaded classes with names matching a search string.

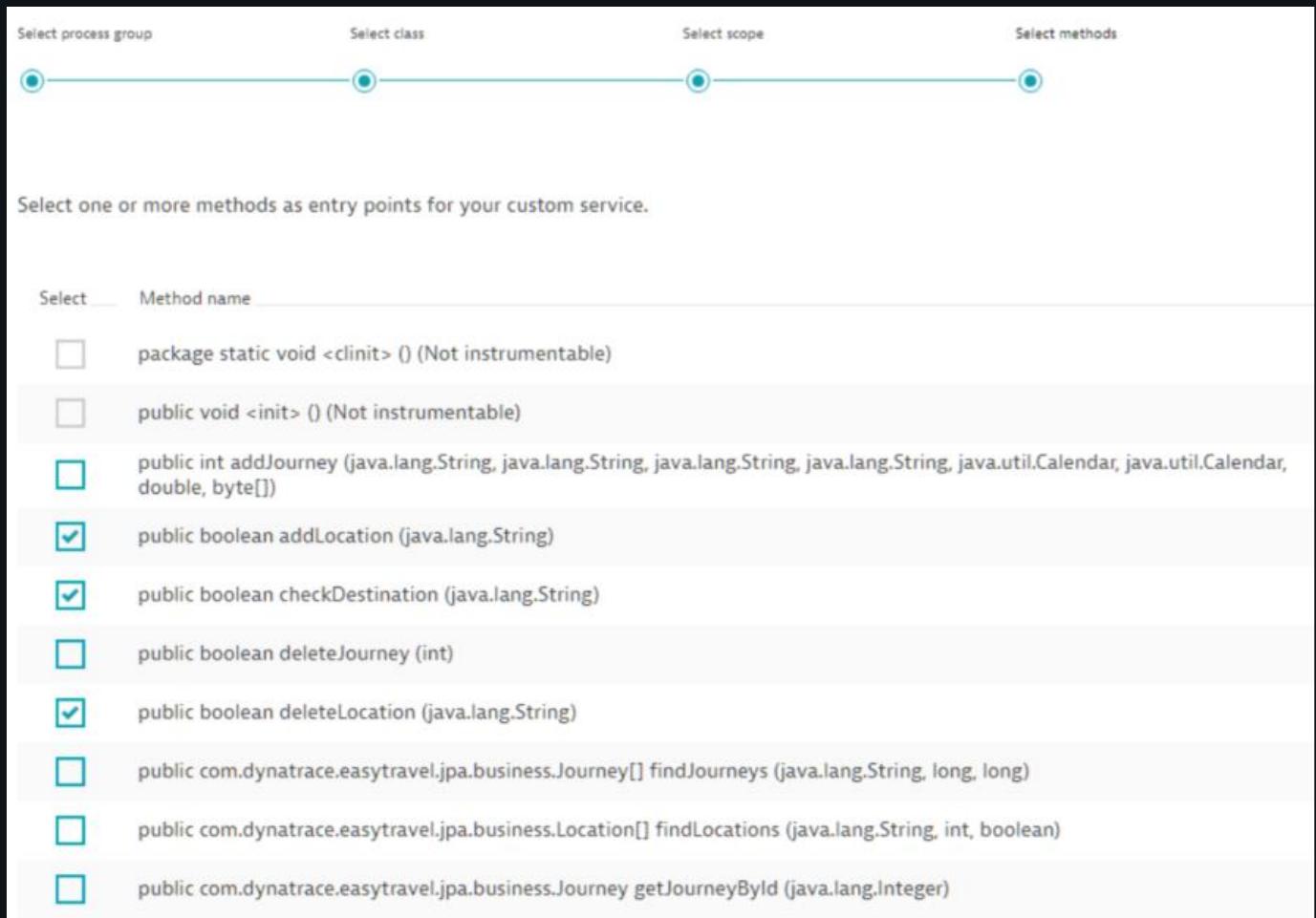
 Search

13 classes found
Select the class that you want to use as a custom service entry point.

- com.dynatrace.easytravel.FindJourneysSqlServerQueryOverride
- com.dynatrace.easytravel.JourneySearchRequestValidator
- com.dynatrace.easytravel.RandomJourneyProvider
- com.dynatrace.easytravel.ScaleMicroJourneyService
- com.dynatrace.easytravel.business.webservice.JourneyService**
- com.dynatrace.easytravel.business.webservice.transferobj.JourneyPage
- com.dynatrace.easytravel.database.JourneyUpdate
- com.dynatrace.easytravel.database.JourneyUpdate\$Mode
- com.dynatrace.easytravel.jpa.business.Journey
- com.dynatrace.easytravel.mongodb.collection.JourneyCollection

Capturing Method Arguments

- Pick one or more methods that you want to capture parameters from



Select one or more methods as entry points for your custom service.

Select	Method name
<input type="checkbox"/>	package static void <clinit> () (Not instrumentable)
<input type="checkbox"/>	public void <init> () (Not instrumentable)
<input type="checkbox"/>	public int addJourney (java.lang.String, java.lang.String, java.lang.String, java.lang.String, java.util.Calendar, java.util.Calendar, double, byte[])
<input checked="" type="checkbox"/>	public boolean addLocation (java.lang.String)
<input checked="" type="checkbox"/>	public boolean checkDestination (java.lang.String)
<input type="checkbox"/>	public boolean deleteJourney (int)
<input checked="" type="checkbox"/>	public boolean deleteLocation (java.lang.String)
<input type="checkbox"/>	public com.dynatrace.easytravel.jpa.business.Journey[] findJourneys (java.lang.String, long, long)
<input type="checkbox"/>	public com.dynatrace.easytravel.jpa.business.Location[] findLocations (java.lang.String, int, boolean)
<input type="checkbox"/>	public com.dynatrace.easytravel.jpa.business.Journey getJourneyById (java.lang.Integer)

Capturing Method Arguments

- Select which argument or return value you want to capture for each method
- Either restart the processes that you want this rule to apply to or ensure that real-time updates is enabled (Java only)

Data sources

Capture Java method parameter(s) from `com.dynatrace.easytravel.business.webservice.JourneyService` globally

Rule applies to requests of the following process group, technology and with process group tag

All process groups

All service technologies

Process group tags

Request attribute source

Java method parameter(s)

Class

`com.dynatrace.easytravel.business.webservice.JourneyService`

Select method sources

Methods

Method Name	Capture	Move up/down	Delete
public boolean addLocation	1: <code>java.lang.String</code>	▼	X
public boolean checkDestination	1: <code>java.lang.String</code>	▲ ▼	X
public boolean deleteLocation	1: <code>java.lang.String</code>	▲ ▼	X
publicJourney[] findJourneys	1: <code>java.lang.String</code>	▲ ▼	X
publicLocation[] findLocations	1: <code>java.lang.String</code>	▲ ▼	X
privateCollection getLocations	1: <code>java.lang.String</code>	▲	X

Search for methods

Deep Object Access

- You can access not only argument values and return values, but also the object itself
- Whenever an item to be captured (be it an argument or an object) is a complex object, you have the ability to define a method (chain) that enables deep access into the object

Capture Java method parameter(s) from com.dynatrace.easytravel.business.webservice.BookingService globally

Rule applies to requests of the following process group, technology and with process group tag

All process groups

All service technologies

Process group tags

Request attribute source

Java method parameter(s)

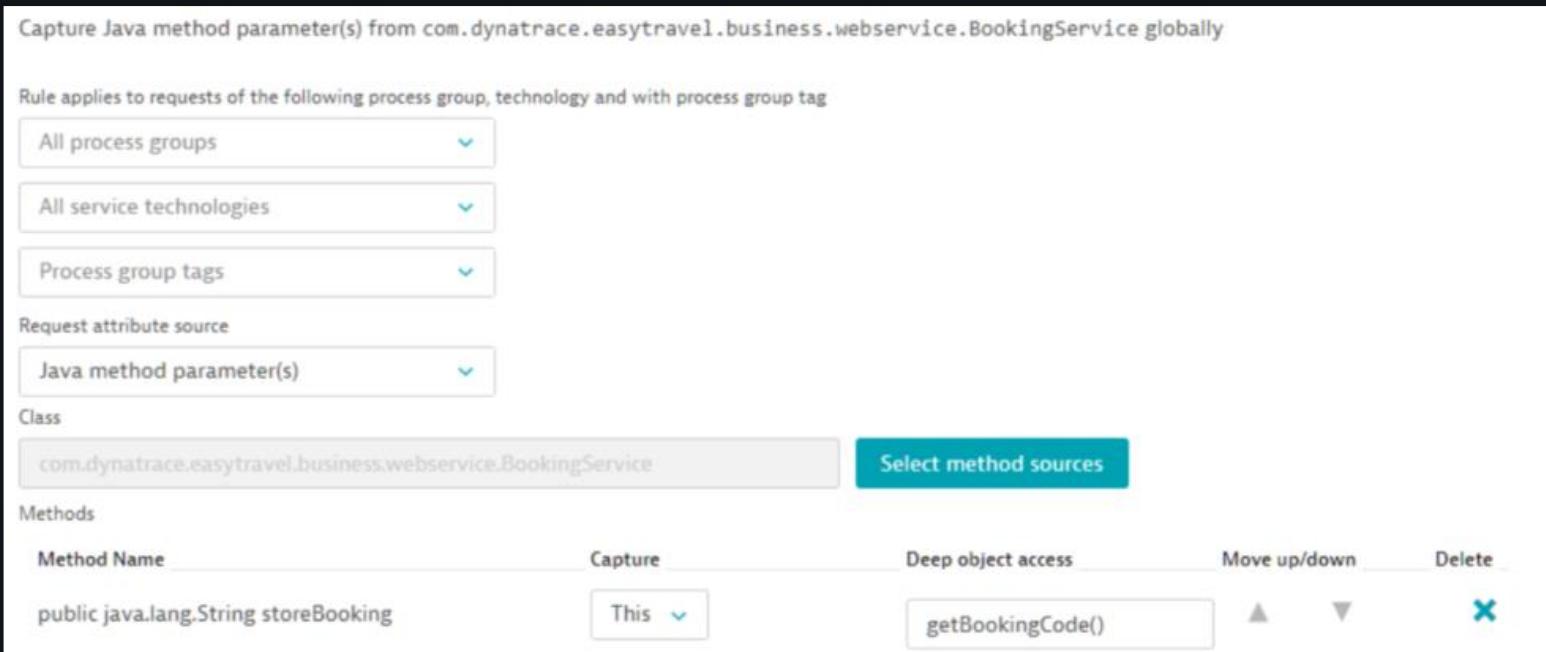
Class

com.dynatrace.easytravel.business.webservice.BookingService

Select method sources

Methods

Method Name	Capture	Deep object access	Move up/down	Delete
public java.lang.String storeBooking	This ▾	getBookingCode()	▲ ▼	X



Viewing Request Attributes

- Once you've defined your attributes, go to any service page where you expect to see your defined request attributes
- Have a look at the Top requests section
- The requests now feature attribute labels indicating that at least some of respective requests contain the new request attribute
- Click any request attribute to filter the entire page view down to only those requests that carry the selected attribute

3 key requests Showing last values

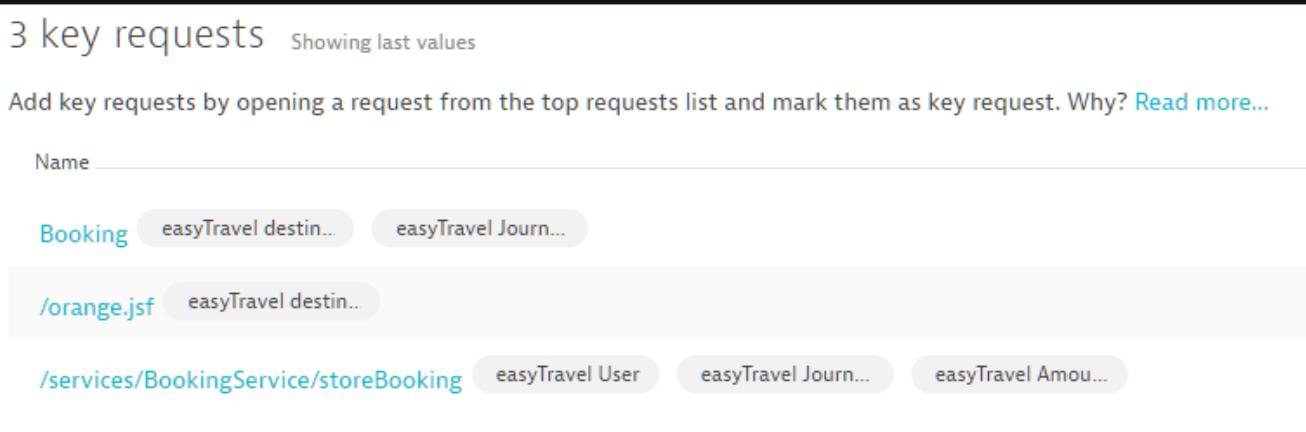
Add key requests by opening a request from the top requests list and mark them as key request. Why? [Read more...](#)

Name

Booking easyTravel destin... easyTravel Journ...

/orange.jsf easyTravel destin...

/services/BookingService/storeBooking easyTravel User easyTravel Journ... easyTravel Amou...



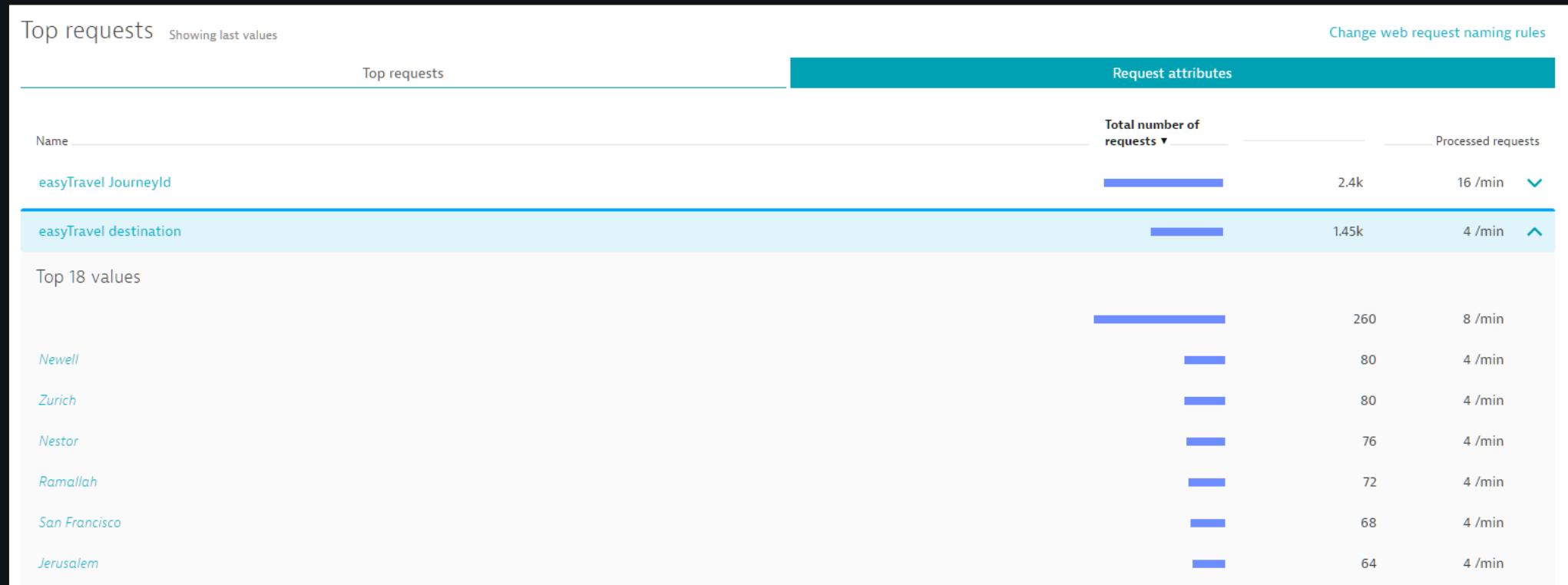
Viewing Request Attributes

- A new Request attributes tab has been added next to the Top requests tab
- This tab lists the request attributes that correspond to the request page
- This table reflects the current filter settings and shows the same metrics as the request table

Top requests <small>Showing last values</small>		Request attributes	
Top requests		Total time consumption ▾	Median response time
Name			
easyTravel JourneyId		<div style="width: 80%; background-color: #6699CC; height: 10px;"></div>	6.5 ms 
easyTravel destination		<div style="width: 40%; background-color: #6699CC; height: 10px;"></div>	217 ms 
easyTravel User		<div style="width: 5%; background-color: #6699CC; height: 10px;"></div>	1.45 s 
easyTravel Amount		<div style="width: 5%; background-color: #6699CC; height: 10px;"></div>	1.45 s 

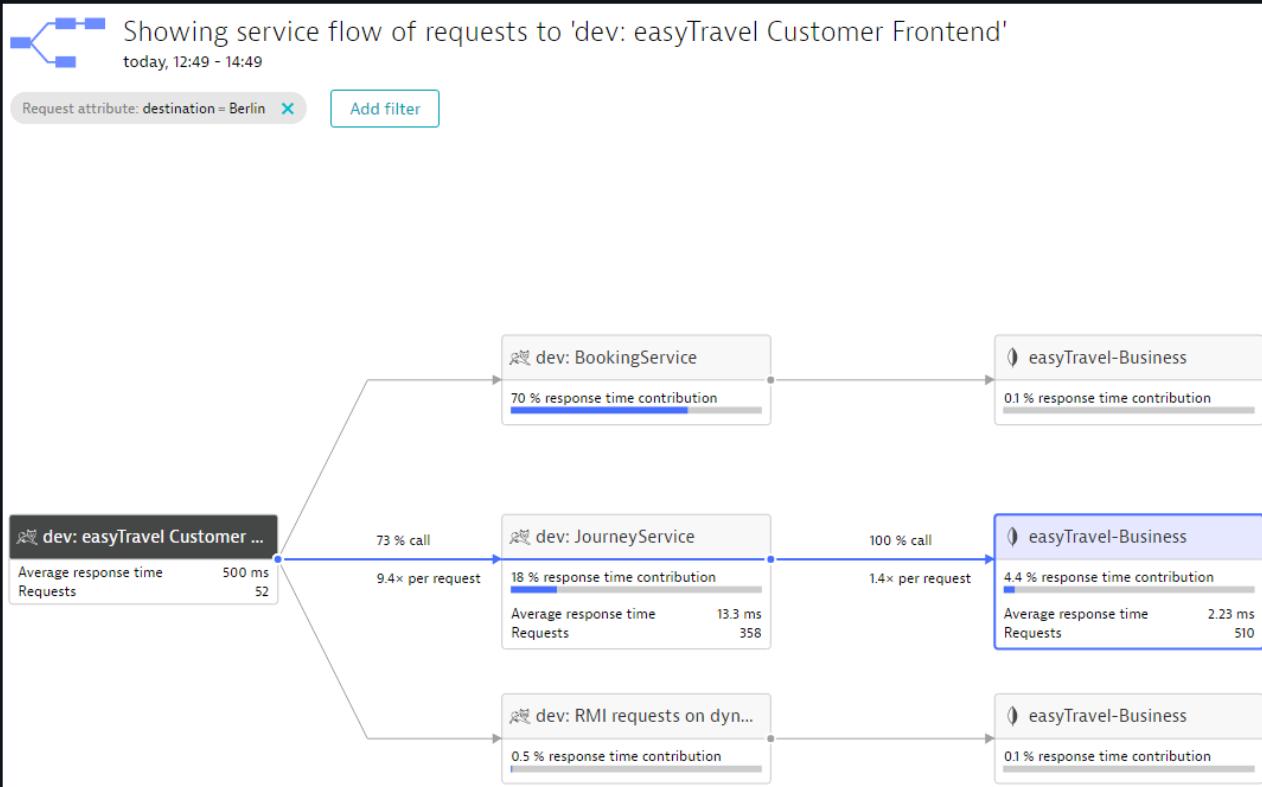
Viewing Request Attributes

- You can see the values by expanding any attribute row



Viewing Request Attributes

- Request attributes can be leveraged across all service analysis views
- The service flow is filtered with the request attribute key/value pair destination = Berlin



Viewing Request Attributes

- Because request attributes can include confidential values, Dynatrace makes it possible to hide sensitive data from certain user groups
- To define or edit a request attribute, users must have the Configure capture of sensitive data permission
- If the confidential data check box is selected for a Request attribute, only users who have the View sensitive request data permission will be able to see the values of the attribute and use the attribute as a filter

The screenshot shows a 'Top requests' dashboard. At the top, it says 'Top requests Showing last values'. Below that, there's a table with two columns: 'Name' and 'Value'. The first row shows 'easyTravel JourneyId' in the 'Name' column and a redacted value '*****' in the 'Value' column. A tooltip 'Top 1 values' is visible near the redacted value. The 'Value' column has a blue header bar.

Request Naming Rules

- You can use request naming rules to adjust how your web requests are tracked and to define business transactions in your customer-facing workflow
- Request Attributes enables you to capture even more context around your requests and to use this to slice and dice through your data

Request Naming Rules

- Define a set of conditions that represent the criteria of your business transaction
- The first condition specifies that:
 - the URL path must contain the string orange-booking
- The second condition specifies that:
 - the request must be an HTTP method of type GET
- The last condition specifies that:
 - the request must have the attribute easyTravel destination.

The screenshot shows a configuration interface for a Request Naming Rule. At the top, there is a toggle switch labeled "Enable rule" which is turned on. Below it, the "Naming pattern" is set to "Booking". The "Condition(s)" section contains three conditions stacked vertically. The first condition is "URL path contains orange-booking". The second condition is "HTTP method equals GET". The third condition is "Request attribute exists easyTravel destination". At the bottom of the interface is a teal-colored button labeled "Add condition" with a white arrow pointing right.

Enable rule

Naming pattern:

Booking

Condition(s):

URL path contains orange-booking

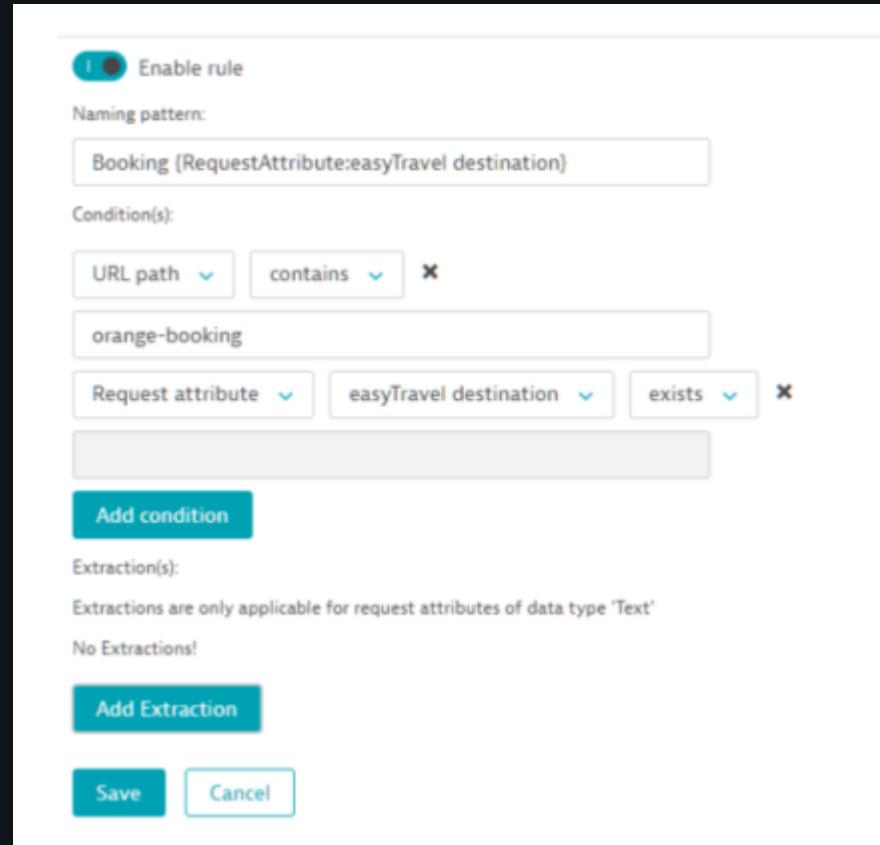
HTTP method equals GET

Request attribute exists easyTravel destination

Add condition

Request Naming Rules

- You may decide to include the value of a request attribute as part of the request name, to create intuitive variant names for this request
- Using the request attribute easyTravel destination, you can create a different request name for each unique value of the attribute



Request Naming Rules

Top requests			
Name	iceform:destinat...	easyTravel desti...	easyTravel Journ...
Start typing to filter table			
/orange.jsf	iceform:destinat...	easyTravel desti...	easyTravel Journ...
Booking Albury	iceform:destinat...	easyTravel desti...	easyTravel Journ...
Booking Nestor	iceform:destinat...	easyTravel desti...	easyTravel Journ...
Booking Stuttgart	iceform:destinat...	easyTravel desti...	easyTravel Journ...
Booking Bugojno	iceform:destinat...	easyTravel desti...	easyTravel Journ...
Booking Miami Beach	iceform:destinat...	easyTravel desti...	easyTravel Journ...
Booking Bagdad	iceform:destinat...	easyTravel desti...	easyTravel Journ...

Showing last PurePaths of requests of

Today, 12:59 - 13:29 (30 Minutes) Apply

contains 'booking' Request attribute: easyTravel destination Add

Filter by name, URI, DB statement

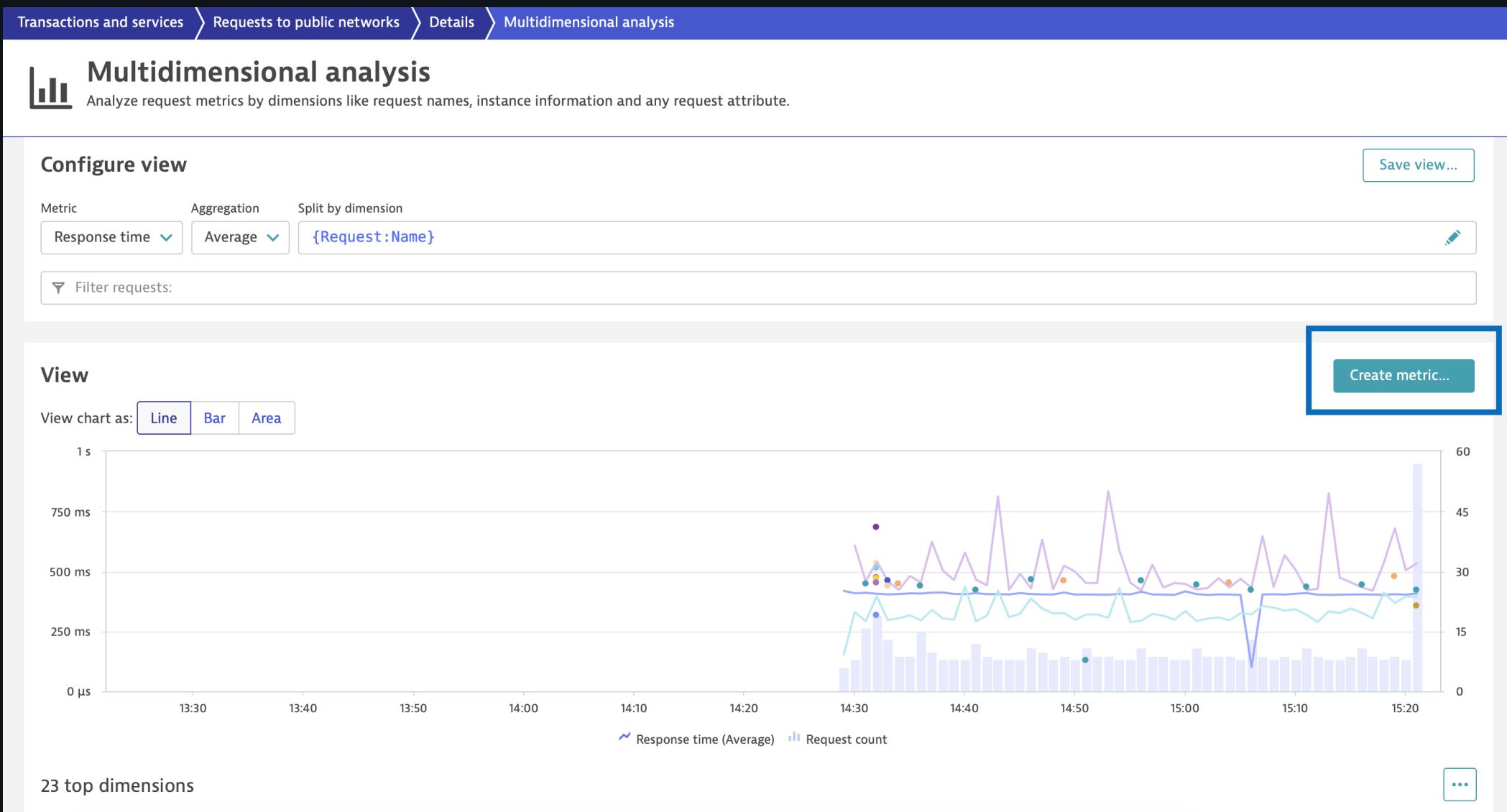
State	Name
	Booking Stuttgart /orange-booking-finish.jsf?journeyId=*****
	Booking Boxford /orange-booking-finish.jsf?journeyId=*****
	Booking Bugojno /orange-booking-finish.jsf?journeyId=*****
	Booking New York /orange-booking-finish.jsf?journeyId=*****
	Booking Bagdad /orange-booking-finish.jsf?journeyId=*****
	Booking Bugojno /orange-booking-finish.jsf?journeyId=*****
	Booking Stuttgart /orange-booking-finish.jsf?journeyId=*****
	Booking New York /orange-booking-finish.jsf?journeyId=*****
	Booking Albury /orange-booking-finish.jsf?journeyId=*****
	Booking Stuttgart /orange-booking-finish.jsf?journeyId=*****
	Booking Berlin /orange-booking-finish.jsf?journeyId=*****

Calculated Service Metrics

Calculated Service Metrics

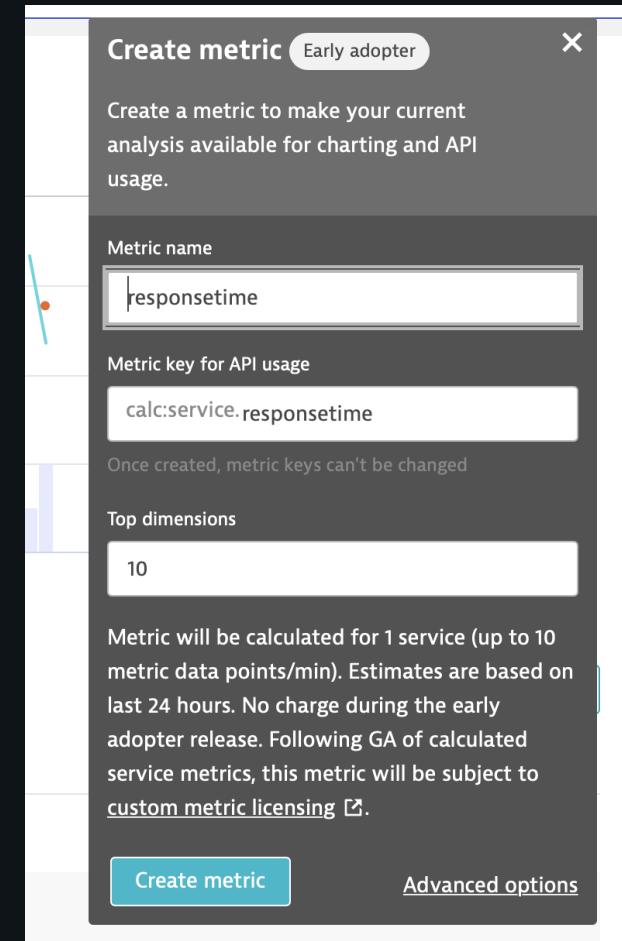
- Calculated Service Metrics allows us to promote captured values from Request Attributes into full-fledged metrics
- This allows us to:
 - Chart the metrics on a dashboard
 - Export the metrics' timeseries and dimensions over the API
 - Set thresholds and alerts based on the metrics
- Calculated service metrics can be created in two ways:
 1. From a multi-dimensional analysis screen.
 2. From the settings menu.

Creating Calculated Service Metrics: From Multi-Dimensional Analysis



Creating Calculated Service Metrics: From Multi-Dimensional Analysis

- Fill in the name of the metric and the metric key
- The Metric key is used internally to store your metric data, and is also used when retrieving metric data over the API.
 - It cannot be changed after metric creation.
- Clicking 'Create Metric' instantly creates your metric based on the filter and dimension defined in the MDA.
- Clicking 'Advanced Options' opens the settings screen.



Creating Calculated Service Metrics: From Multi-Dimensional Analysis

The screenshot shows the 'Create metric' page in the Dynatrace interface. At the top, there's a breadcrumb navigation: 'Calculated service metrics > Create metric'. Below it, a section titled 'Calculated service metrics' includes an 'Early Adopter' badge. A note states: 'Dynatrace automatically captures important metrics for services with no configuration required.' with a 'More...' link. The main form starts with a 'Naming' section where users can enter a 'Metric name' ('responsetime') and a 'Metric key for API usage' ('responsetime'). A note below says: 'Once created, metric keys can't be changed.' Next is a 'Metric source' section with dropdowns for 'Response time' and 'Unit' ('Microsecond'). At the bottom, a 'Service' section is highlighted with a blue border, containing the text 'Requests to public networks'.

Please note that the service cannot be changed if you create the metric this way

Creating Calculated Service Metrics: From Settings page

- Name the metric and provide the metric key
- Select the metric source
- When selecting a built-in metric source, the unit that Dynatrace uses is selected for you.

Calculated service metrics Early Adopter

Dynatrace automatically captures important metrics for services with no configuration required. [More...](#)

Naming
Name your service metric and give it an unique key.

Metric name	Metric key for API usage
Responsetime	responsetime

Once created, metric keys can't be changed.

Metric source

Response time	Unit
Response time	Microsecond
Request count	
Successful request count	
Failed request count	
Exception count	
Failure rate	
HTTP 4xx count	

Successful request count

sets it should take into account when calculating the new metric. Each service that fits the conditions will receive a time series for this metric.

Creating Calculated Service Metrics: From Settings page

- When you want to use a request attribute, you can only use non-confidential attributes.
- You also have to select the unit, what your metric represents

Calculated service metrics Early Adopter

Dynatrace automatically captures important metrics for services with no configuration required. [More...](#)

Naming
Name your service metric and give it an unique key.

Metric name	Metric key for API usage
<input type="text" value="Responsetime"/>	<input type="text" value="responsetime"/>
Once created, metric keys can't be changed.	

Metric source
 Contains only non-confidential request attributes

Unit
 Unit not applicable

 Cores

 Unit not applicable

Management zone

Conditions
The set of conditions that tell Dynatrace which requests it should take into account when calculating the new metric. Each service that fits the conditions will receive a time series for this metric.



Creating Calculated Service Metrics: From Settings page

- A metric can belong to at most 100 services.
To reach this number, you must apply at least one filter (even if your environment has less than 100 services)
- You must provide a filter to limit your metric.
This can be a combination of:
 - Management zone
 - Service Tag
 - Service Name
 - Actor system
 - Azure function function name
 - Azure function site name
 - CTG gateway URL
 - CTG server name
 - ESB application name
 - Public domain name
 - Remote endpoint
 - Remote service name
 - Web application id
 - Web context root
 - Web server name
 - Web service name
 - Web service namespace

Creating Calculated Service Metrics: Conditions

Conditions

The set of conditions that tell Dynatrace which requests it should take into account when calculating the new metric. Each service that fits the conditions will receive a time series for this metric.

Add condition

Condition	Delete	Edit
Service display name contains 'Requests to public' case insensitive	X	▼

Metric will be calculated for 1 services and produces up to 1 data points/min (estimation based on the last 24 hours).
No charge during the early adopter release. Following GA of the calculated service metrics, this metrics will be subject to [custom metric licensing](#).

Service ▲	Process group ▲
Requests to public networks	

Once you set a filter for certain services, the list of services this metric would apply to is displayed

Creating Calculated Service Metrics: Conditions

- The conditions screen can also be used to filter requests inside services.
- Examples:
 - Requests that take longer than 10 seconds
 - Requests with a certain URL pattern
 - Requests where the attribute 'amount' is larger than 100

Conditions

The set of conditions that tell Dynatrace which requests it should take into account when calculating the new metric. Each service that fits the conditions will receive a time series for this metric.

Add condition

Request attribute UserID

Case sensitive

Add **Cancel**

Condition	Delete	Edit
Service display name contains 'Requests to public' case insensitive	X	▼

Creating Calculated Service Metrics: Dimensions

- Finally, we can opt to split our metric into dimensions
- This is not always necessary if I want to see one final number for a service (e.g. total #products ordered), I do not need a dimension as the default metric will display the total number of ordered products.
- If I want to see the total number of products per country, I would create a dimension and split on

Split by dimension

Dimensions allow you encode additional information for each captured value of your metric, for example, response times per HTTP request type. [More...](#)

Dimension value pattern

{RequestAttribute:Country}

Use {} to add predefined or custom placeholders to your dimension value.

Dimension name

Country

Number of top values Value sorting Value aggregation

10

Highest to lowest ▾

Sum of values ▾

Add custom placeholder

Calculated Service Metrics

- Calculated Service Metrics can be accessed in the 'Service' portion when creating a custom chart, or in the v2 version of the Environment API (metrics endpoint)



Limitations

- Creating a metric directly from a multidimensional analysis view, binds the metric to that service.
 - If the underlying process group changes (e.g. different hostgroup), the service changes and the metric will no longer collect data.
 - For this reason, we recommend creating them from the settings screen.
- Out-of-the box, you can:
 - Create a total of 500 calculated service metrics
 - Every metric can apply to a maximum of 100 services
 - A metric can have a maximum of 100 dimensions per service.
 - A metric with 100 dimensions over 100 services, would have a maximum of 10.000 different datapoints.

API Detection Settings

API Detection

Settings > Server-side service monitoring > API detection rules

API detection rules

Modern applications use a lot of different frameworks, so stacktraces in method hotspots and exceptions can become quite long. APIs allow you to track component and the respective ownership that is responsible.

Create API detection rule

User-defined APIs

Filter by name

Name	Move up/down	Delete	Edit	Details
Derby	▼	X	/	/
Jmx	▲ ▼	X	/	/
Tomcat	▲ ▼	X	/	/
XML Processing	▲ ▼	X	/	/
Agent	▲ ▼	X	/	/
HornetQ	▲ ▼	X	/	/
easyTravel	▲	X	/	/

Built in API Rules

Built-in APIs

Name	Details
Apache	^

Matches if the **fully qualified class name begins with** 'org.apache.'

Create a new API definition

The screenshot shows the 'API detection rules' configuration page in the Dynatrace interface. A blue callout bubble labeled 'Rule Name' points to the 'API display name' input field, which contains 'Easytravel Business'. Another blue callout bubble labeled 'Conditions' points to the 'Add new condition' button and the first condition listed in the table.

Settings > Server-side service monitoring > API detection rules

Settings

Monitoring

- Setup and overview
- Monitored technologies
- Monitoring overview
- Host naming

Process groups

- Detection and naming

Web & mobile monitoring

- Real user & synthetic monitoring

Cloud and virtualization

- Connect vCenter, Azure, Cloud Foundry ...

Server-side service monitoring

- Manage & customize service monitoring
- Custom service detection
- Merged service monitoring
- Service naming rules
- Request attributes

Add API detection rule

Rule Name

API display name

Easytravel Business

Color

Add new condition

Conditions

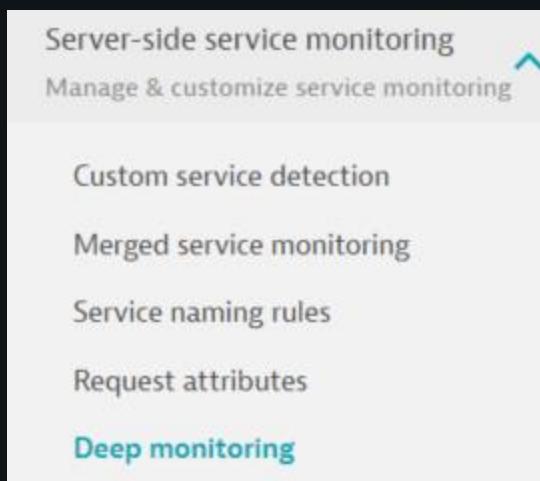
Condition	Delete	Edit
Matches if the fully qualified class name begins with ''.	X	^
<p>Base</p> <p>Fully qualified class name (Java, .NET, ...)</p> <p>Matcher</p> <p>begins with</p> <p>Pattern</p> <p>com.dynatrace.easytravel.business</p>		

Confirm Cancel

Deep Monitoring

Deep Monitoring

- Dynatrace OneAgent automatically traces each detected transaction end-to-end, capturing all related WebRequest calls, database calls, remoting calls, exceptions, and more
- Use the settings here to exclude certain URLs or exceptions from capture to reduce network bandwidth, storage requirements, misleading alerts, or other monitoring noise
- The troubleshooting settings enable you to disable certain OneAgent functionality at a fine-grained level to facilitate issue resolution when working with Dynatrace Support



Deep Monitoring

Exclude noisy and unnecessary exceptions

Exclude useless and noisy exceptions from being captured in monitored requests to reduce network bandwidth, storage requirements and potential overhead. Dynatrace will still try capture these exceptions if they are immediate cause of a failing request.

Java exception rules

.NET exception rules

Add Java exception exclusion rule

Java exception exclusion rule

Delete Details

Don't capture Java exceptions of any class that **begins with** 'com.bea.'

built-in

Don't capture Java exceptions of any class that **begins with** 'com.guidewire.'

built-in

Don't capture Java exceptions of any class that **begins with** 'com.ibm.'

built-in

Don't capture Java exceptions of any class that **begins with** 'com.netscape.'

built-in

Don't capture Java exceptions of any class that **begins with** 'com.octetstring.'

built-in

Add Java exception exclusion rule

Don't capture Java exceptions of any class that **begins with**

and whose message **begins with** **optional**

Deep Monitoring

Exclude specific web request URLs

Exclude useless and noisy web requests from being monitored. These requests are then neither captured, monitored nor alerted upon.

[Add web request URL exclusion rule](#)

Web request URL exclusion rule	Delete	Details
No web request URL exclusion rules defined		

Add web request URL exclusion rule

Don't monitor web requests with URL path that begins with and whose query begins with optional

Deep Monitoring

▼ Troubleshooting requires OneAgent 1.145+

These settings are meant to be used together with the Dynatrace support team to help troubleshooting support tickets related to the OneAgent. It should not be used without guidance. Switching instrumentation on or off requires an agent restart.

Capture PurePath On

Capture SQL Select row count On

Capture method hotspot information in PurePaths On

[Disable specific Sensor](#)

Disabled Sensors	State	Delete	Details
No sensors disabled			

[Troubleshoot a specific process group](#)

Process group troubleshooting settings	Delete	Edit
No process group troubleshooting settings defined		

[Reset to default](#)

Deep Monitoring

- Bind Values

Managed Only

The screenshot shows the 'Database' section of the Deep Monitoring configuration. It includes a 'Beta' status indicator and a note about OneAgent requirements. A blue speech bubble with the text '**Managed Only**' has an arrow pointing to the 'Capture SQL bind values' toggle switch. Below the main section is a button to add database settings for specific process groups, followed by a 'Filter for...' dropdown and a table row for 'Process group database settings'. The table row contains columns for 'Edit', 'Delete', and a message stating 'No process group settings defined'.

Database Beta requires OneAgent 1.151+

Enable capturing of SQL bind variables. May result in additional traffic and storage overhead.

Capture SQL bind values

Add database settings for a specific process group

Filter for...

Process group database settings	Delete	Edit
No process group settings defined		

Adaptive Capture Control

Control the number of entry points captured per Environment

The screenshot shows the Dynatrace interface under the 'Server-side service monitoring' section. A blue callout box highlights the 'Deep monitoring' section on the left sidebar, which is currently selected. The main content area displays the 'Adaptive capture control' settings. It includes a descriptive text about how adaptive capture control facilitates end-to-end distributed tracing by capturing PurePaths. Below this is a slider for overriding tenant defaults, set at 1000 PurePaths/min. A button labeled 'Add adaptive capture control settings for a specific process group' is present, along with a note that no process group activity has occurred in the last 72 hours. A 'Filter for...' dropdown is also shown. At the bottom, a message states 'No process group settings defined'. The sidebar also lists other monitoring features like Log Analytics, Anomaly detection, Alerting, Integration, Tags, and Maintenance.

Adaptive capture control

Each deep monitored process is capturing PurePaths to facilitate end to end distributed tracing. The adaptive capture control limit specifies how many entry point traces are being captured by any given process. If a process starts more traces than this limit adaptive capture control will select a suitable sample set. This logic is described [here](#) and ensures that a statistical valid sub set of the overall number of requests are captured. It also ensures that the OneAgent always captures at least that number and never less.

Override tenant default (1000 PurePaths/min)

Number of new PurePaths traced per process and minute

1000 PurePaths/min

Add adaptive capture control settings for a specific process group

No process group activity in the last 72 hours

Filter for...

Process group adaptive capture control settings

Edit

No process group settings defined

OneAgent beta features

Enable OneAgent features that are currently in public beta.

Database Beta requires OneAgent 1.151+

Enable capturing of SQL bind variables. May result in additional traffic and storage overhead.

Questions?



Simply smarter clouds