



# Transaction Snapshots

- [Deep Code-Level Visibility in Production](#)
- [Overview of Transaction Snapshots](#)
  - [To view transaction snapshots](#)
  - [Transaction Snapshot Views](#)
    - [Snapshot Flow Map](#)
    - [Snapshot Waterfall View](#)
    - [The Snapshot List View](#)
  - [Transaction Snapshot Call Drill Downs](#)
  - [Diagnostic Data Captured by a Transaction Snapshot in the Call Drill Down Window](#)
- [Using Transaction Snapshots](#)
  - [Sorting and Searching for Specific Transaction Snapshots](#)
    - [To filter transaction snapshots using search criteria](#)
    - [To filter transaction snapshots by refining the results list](#)
  - [Comparing Snapshots](#)
    - [To compare snapshots](#)
  - [Troubleshooting With Snapshots](#)
    - [To analyze the most expensive calls and SQL statements](#)
  - [Archiving Snapshots](#)
    - [To Archive a Transaction Snapshot](#)
    - [To Find Archived Transaction Snapshots](#)
- [Learn More](#)

## Expert Advice

### **The Intelligent Approach to Production Monitoring**

*by Jim Hirschauer*

A transaction snapshot depicts a set of diagnostic data, taken at a certain point in time, for an individual transaction across all app servers through which the transaction has passed.

Transaction snapshots are the vehicle for troubleshooting the root causes of performance problems.

## Deep Code-Level Visibility in Production

Code level visibility is essential for troubleshooting performance problems in production. You need details about the exact code path taken by a particular transaction and the time spent in the methods that were executed. A distributed environment presents a challenge because multiple code paths are executed across multiple application servers.

## Overview of Transaction Snapshots

AppDynamics generates a transaction snapshot to capture the code paths executed on instrumented application servers involved in a distributed transaction.

See how AppDynamics monitored performance in real time during elections

*by Sandy Mappic*

Transaction snapshots contain considerable amounts of data and are processed and sent from the app server on which the application is running. Therefore, they are captured selectively under following conditions:

- **When a diagnostic session is triggered**  
AppDynamics starts diagnostic sessions when it detects a pattern of performance problems. In addition you can manually start a diagnostic session from the Business Transaction Dashboard. For details see [Diagnostic Sessions](#).
- **When slow, stalled, or error transactions are identified**  
These snapshots may have partial call graph information, starting at the time when the transaction slowed or experienced an error.
- **Based on the Periodic Collection setting**  
By default AppDynamics captures one snapshot every 10 minutes. For details see [Configure Transaction Snapshots](#).

### To view transaction snapshots

You can get a list of transaction snapshots for the selected time range:

- From the **Transaction Snapshots** tab of the application, tier, node, or business transaction dashboards
- From the links in the transaction scorecards in the application, tier, node, or business transaction dashboards
- From the **Troubleshoot-> Slow Response Time** or **Troubleshoot-> Errors** in the left navigation pane

1. Navigate to an application, tier, node, or business transaction dashboard.
2. Click the **Transaction Snapshots** tab.
3. From the list of transaction snapshots that displays, select the snapshot that you want to view and click **View Transaction Snapshot**.
4. In the transaction flow map click **Drill Down**.

Or

1. Navigate to an application, tier, node, or business transaction dashboard.
2. Click the individual slow, very slow, stalls or errors links in the Transaction Scorecard section of the dashboard.
3. From the list of transaction snapshots that displays, select the snapshot that you want to view and click **View Transaction Snapshot**.
4. In the transaction flow map click **Drill Down**.

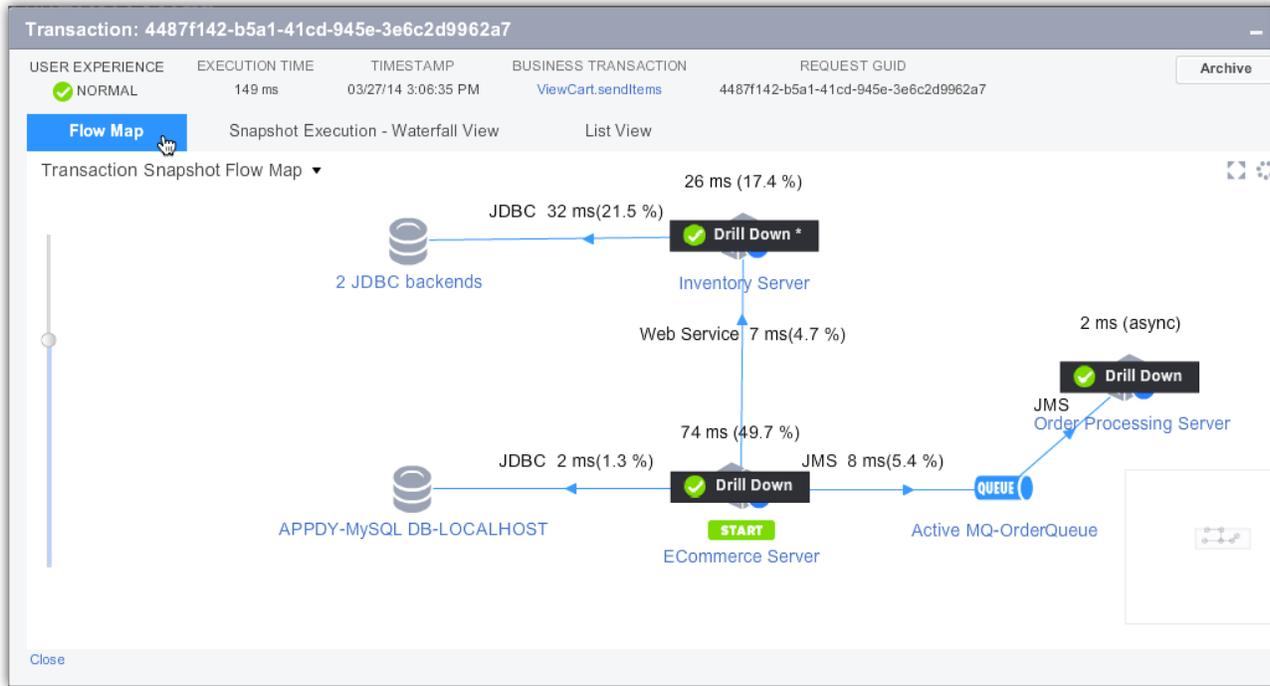
Or

1. Click **Troubleshoot-> Slow Response Time** or **Troubleshoot-> Errors** in the left navigation pane.
2. Select a slow or error transaction from the list.
3. Click **View Transaction Snapshot**.
4. In the transaction flow map click **Drill Down**.

## Transaction Snapshot Views

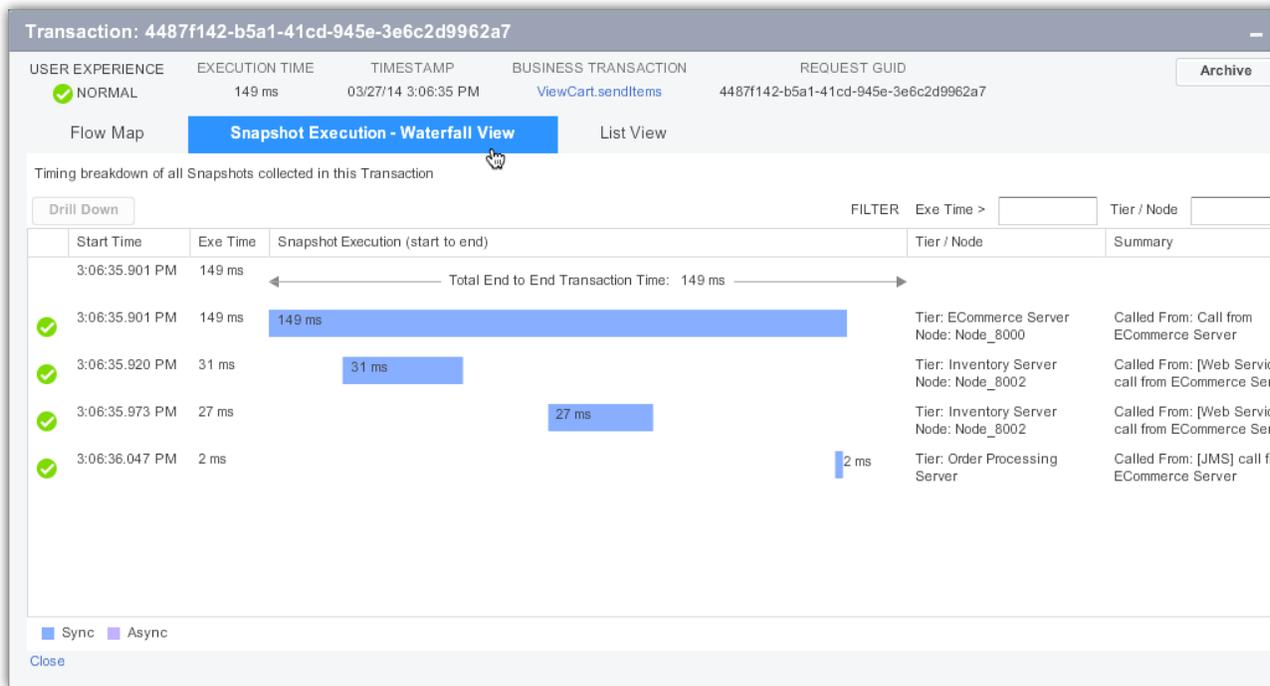
### Snapshot Flow Map

When you double-click on any item in the list of transaction snapshots, a Transaction Snapshot Flow Map displays. You can see the user experience, execution time, and timestamp of the transaction. The flow map also provides details of the overall time that is spent in a particular tier and in database and remote service calls.

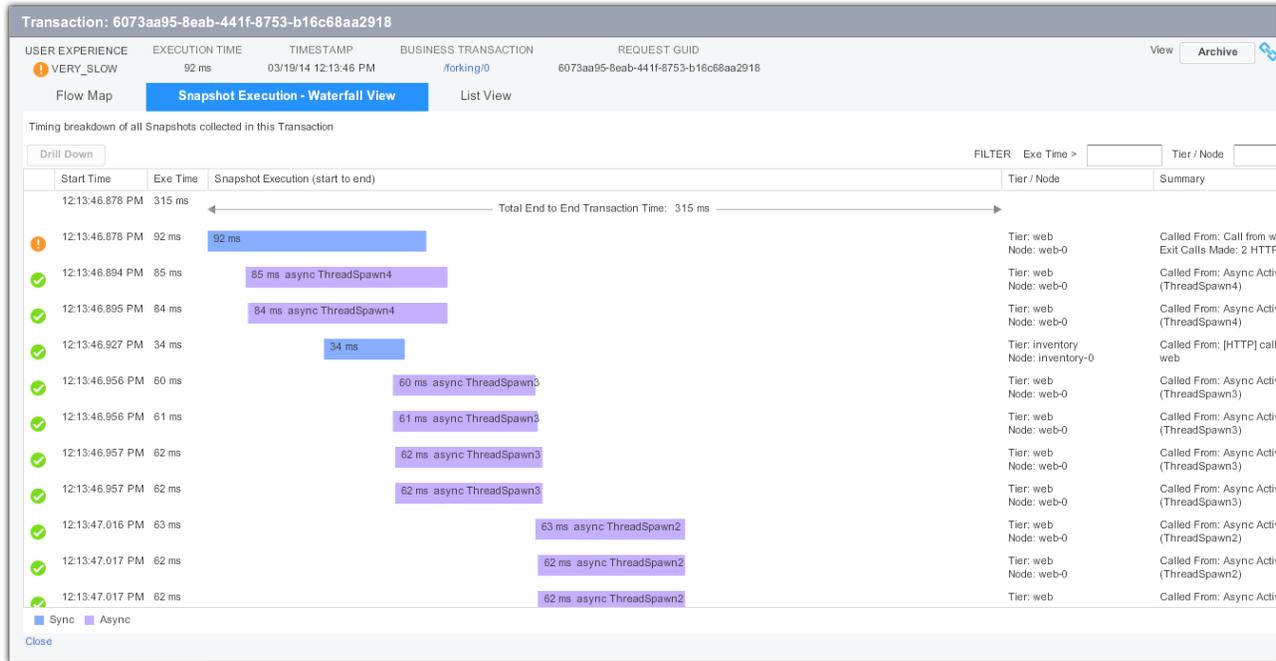


### Snapshot Waterfall View

When you click the Snapshot Waterfall View tab, you see a timing waterfall chart of the snapshots collected for a business transaction. The chart visualizes the call execution times as they occur during the end-to-end transaction time.

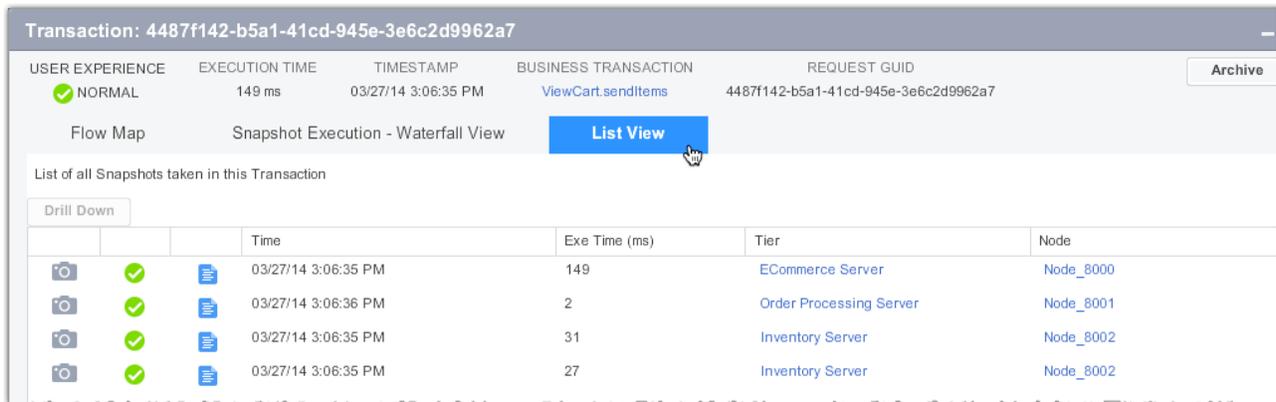


If there is asynchronous activity the waterfall view displays those calls in a different color.



## The Snapshot List View

The Snapshot List View shows a simple list. The list is very useful for sorting snapshots according to execution time.



## Transaction Snapshot Call Drill Downs

An individual transaction snapshot contains diagnostic information for individual business transaction instances. It provides information to help diagnose why a particular transaction or series of transactions are not performing as well as expected.

To get the call drill down information:

- In the Transaction Snapshot Flow Map view, click **Drill Down** on a tier.
- In the Snapshot Waterfall View select a snapshot and double-click or click **Drill Down**.
- In the Snapshot List View select a snapshot and double-click or click **Drill Down**.

**i** The contents of a transaction snapshot containing async segments look slightly different if you access the snapshot via the Business Transaction view or via the App/Tier/Server view. In the Business Transaction view, only the originating segments are shown initially, and then you can drill down to the async segments as desired. Because the App/Tier/Server view surfaces all the segments that are relative to that entity, all segments, originating or async, are listed initially.

## Diagnostic Data Captured by a Transaction Snapshot in the Call Drill Down Window

The following details are captured in a transaction snapshot:

- **Summary:** Problem summary, execution time, CPU, timestamps tier, node process ID, thread name, URL, Session ID, etc.
- **Call Graphs:** Call graphs for every tier involved in a transaction. You can drill down to the method call that is causing the problem. AppDynamics automatically filters non-application classes such as core Java classes, third party libraries, application server and database driver implementation classes. You can configure call graph settings to control which classes should be included or excluded from the call graph. To configure, see [Configure Call Graphs](#).

- **Hot Spots:** Sorts the calls by execution time, with the most expensive calls at the top. To see the invocation trace of a single call in the lower panel, select the call in the upper panel. Use the slider to filter which calls to display as hot spots. For example, the following setting

filters out all calls faster than 30 ms from the hot spots list.



Using the [force-hotspot-if-diag-session](#) and [hotspot-collect-cpu](#) node properties you can respectively control whether or not hot spot snapshots are collected for manually started diagnostic sessions and whether CPU time or real time is collected within the hot spot snapshots.

- **SQL Calls:** All SQL queries fired during a request. AppDynamics normalizes the queries and by default does not display raw/bind values. You can configure SQL capture settings to monitor raw SQL data in the queries. Calls taking less than 10 ms are not reported but when they occur multiple times you may see <SQL Call(s)> in the Query column to identify these calls. To configure, see [Configure Call Graphs](#).

Query Type	Query	Avg. Time (ms)	Count	Tot. Time (ms)	% Time	From Tier	To Tier	Error
Query	SELECT COUNT(1) COUNT FROM ITEM IT1, ITEM IT2	16052	1	16052	99.8	E-Commerce-2		
DB Transaction Commit	COMMIT	5	1	5	0.0	E-Commerce-2		
Delete	delete from cart	1	1	1	0.1	E-Commerce-2		
4 X [SELECT CATFILTER.CATFILTER_ID, CATFILTER.IDENTIFIER, CATFILTER.STOREENT_ID,		0	0	0	0.0	E-Commerce-2		
Get Pooled Connection From DataSource		0	2	0	0.0	E-Commerce-2		
select this_id as id1_0, this_title as title1_0, this_imagePath as imagePath1_0, from item this_		0	1	0	0.0	E-Commerce-2		

- **HTTP Params:** HTTP payloads contain basic data such as the URL, session ID, and User Principal (which are displayed in the SUMMARY panel) and additional data for Servlet entry points, Struts, JSF, Web Services, etc. To help identify user and server state at the time a snapshot was taken, you can create HTTP data collectors to specify query parameters, headers, and session keys to be captured in the transaction snapshot. To configure, see [Configuring HTTP Request Data Collectors](#).
- **Cookies:** The snapshot can use cookie values to help identify the user who initiated the slow or error transaction. To configure, see [Configuring HTTP Request Data Collectors](#).
- **User Data:** User data from any method executed during a transaction, including parameter values and return values, to add context to the transaction. You can use method invocation data collectors to specify the method and parameter index. To configure, see [Configuring Method Invocation Data Collectors](#).
  - ⚠ In cases where an exit call is made just before a business transaction starts, exit call information can show up in this field, particularly if the transaction is marked as slow or having errors. Please note that sensitive information on the exit call may be shown in this situation.
- **Error Details:** Exception stack traces and HTTP error codes.
- **Hardware/Mem:** Graphs for hardware (CPU Memory, Disk IO, Network IO), Memory (Heap, Garbage Collection, Memory Pools), JMX, etc.
- **Node Problems:** Viewer to analyze node problems. See [Troubleshoot Node Problems](#).

- **Additional Data**

Transaction snapshots include distributed call graphs and response time distribution details only when a series of bad transactions or a performance policy violation trigger a diagnostic session on a node.

## Using Transaction Snapshots

### Sorting and Searching for Specific Transaction Snapshots

#### To filter transaction snapshots using search criteria

1. In the **Transaction Snapshots** tab click the **All Snapshots** subtab.
2. Click **Show Filters** if filters are not showing.
3. Click **Search Criteria**. Check the following criteria to sort the list:
  - **User Experience**
    - Slow
    - Very Slow
    - Stall
  - **Execution Time** in milliseconds
  - **Business Transaction**
    - Click **Add** (the + icon) to select a Business Transaction as a search criteria
  - **Errors**
    - Click **Error Occurred** to list all errors
    - Click **Add** (the + icon) to select a particular error as a search criteria
  - **HTTP Request Data**
    - URL
    - Session ID
    - User Principal
  - **Data Collector Data**
    - Collector Type: Any, HTTP Parameter, Business Data, Cookie
    - Name
    - Value
  - **Archived**
    - Return Only Archived Snapshots

- **Advanced**
  - Request GUIDs

3. Click **Search**.

### To filter transaction snapshots by refining the results list

You can refine the results list of a previous query. AppDynamics instantly updates the list as you select or deselect the **Refine Results** filters.

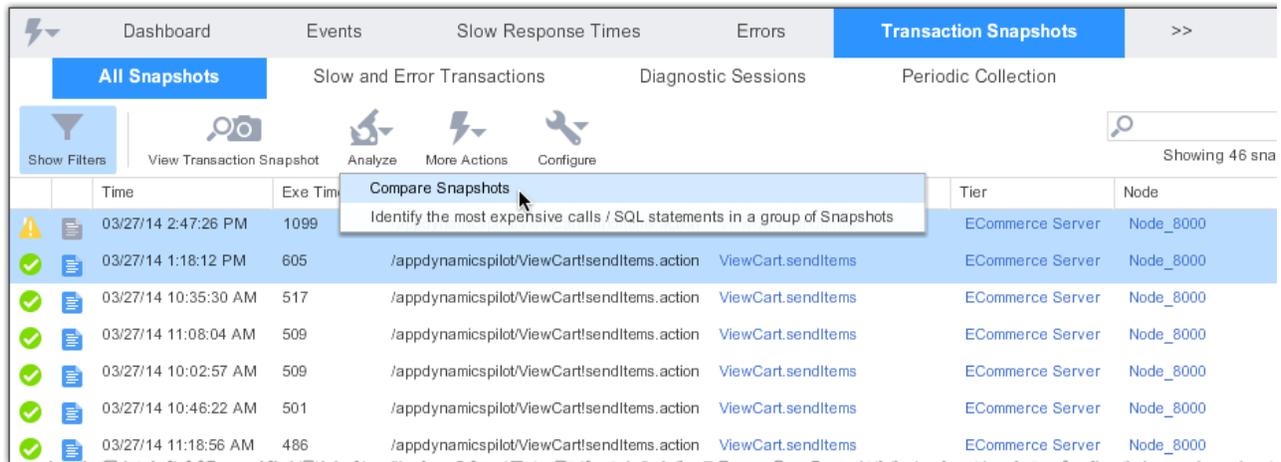
1. In the **Transaction Snapshots** tab of a dashboard click the **All Snapshots** subtab.
2. Click **Show Filters**.
3. Click **Refine Results**. Use the following criteria to sort the list:

- **User Experience**
  - Normal
  - Slow
  - Very Slow
  - Stall
- **Business Transactions**
  - Business Transactions by name  
AppDynamics also shows the number of snapshots.
- **Tiers**
  - Tiers by name  
AppDynamics also shows the number of snapshots.
- **Nodes**
  - Nodes by name  
AppDynamics also shows the number of snapshots.
- **Errors**
  - Snapshots where an error occurred

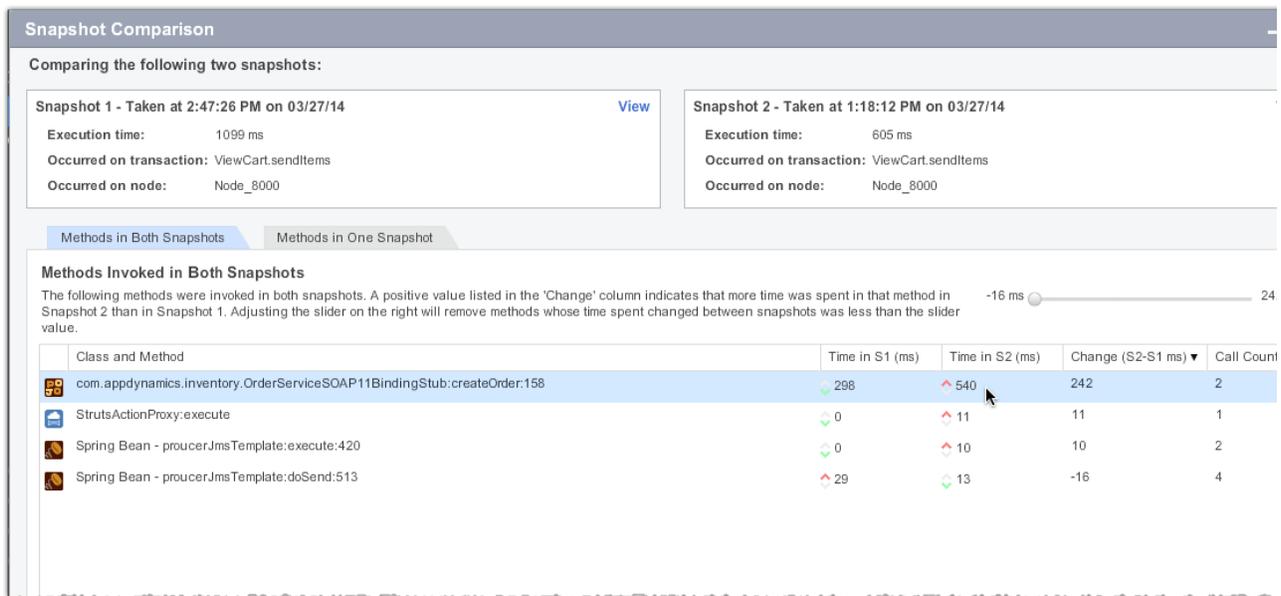
### Comparing Snapshots

#### To compare snapshots

1. In the Business Transactions All Snapshots list, select two snapshots that you want to compare.
2. Click **Analyze -> Compare Snapshots**.



The Snapshot Comparison window displays a comparison of the selected snapshots. Look at the Time and Change columns to find slow methods.



## Troubleshooting With Snapshots

### To analyze the most expensive calls and SQL statements

1. In the Business Transactions All Snapshots list, select a single or a group of snapshots that you want to analyze. You can select up to 30 snapshots.

## 2. Click **Analyze** -> **Identify the most expensive calls / SQL statements in a group of snapshots.**

AppDynamics displays the most expensive calls (methods) in the snapshots and the most expensive SQL statements.

Time	Exe Time (ms)	URL	Business Transaction	Tier	Node
03/27/14 1:40:15 PM	2285	/appdynamicsplot/ViewCart!sendItems.action	ViewCart.sendItems	ECommerce Server	Node_8000
03/27/14 1:39:53 PM	141	/appdynamicsplot/ViewCart!sendItems.action	ViewCart.sendItems	ECommerce Server	Node_8000
03/27/14 1:38:25 PM	1824	/appdynamicsplot/ViewCart!sendItems.action	ViewCart.sendItems	ECommerce Server	Node_8000
03/27/14 1:38:25 PM	1852	/appdynamicsplot/ViewCart!sendItems.action	ViewCart.sendItems	ECommerce Server	Node_8000

Class and Method	Total Exe. Time(ms)	Avg. Exe. Time(ms)	Call Count
com.appdynamics.inventory.OrderServiceSOAP11BindingStub.createOrder:158	1660	103.8	16
Spring Bean - cartPersistence.getCartSize:41	172	43.0	4
Spring Bean - itemPersistence.getItemById:24	152	38.0	4
Spring Bean - proucerJmsTemplate.doSend:513	104	5.2	20
Constructor of com.appdynamics.inventory.OrderServiceLocator:12	104	20.8	5
Spring Bean - transactionManager.doBegin:510	89	22.3	4
Spring Bean - cartServiceTarget.checkOut:40	81	6.8	12
Proxy For Spring Bean - cartServiceTarget.checkOut	50	5.6	9
com.appdynamics.jdbc.MPreparedStatement.executeQuery:45	32	16.0	2
Spring Bean - proucerJmsTemplate.execute:430	22	2.2	10

## Archiving Snapshots

Normally transaction snapshots are purged after a configurable time - by default, two weeks. To save a snapshot beyond the normal snapshot lifespan – for example, if you want to make sure a snapshot associated with a particular problem is retained for future analysis – archive the snapshot.

Customers with on-premise controllers can modify the default two-week period by configuring the snapshots.retention.period in the Controller Settings section of the Administration console.

### To Archive a Transaction Snapshot

1. Display the Transaction Snapshot flow map.
2. Click the **Archive** button in the upper right corner.

When you are viewing your snapshot list, a small icon in the far right column indicates that a snapshot has been archived.

## To Find Archived Transaction Snapshots

To display only archived snapshots in the snapshot list, filter the snapshot list and check **Return Only Archived Snapshots**. See [To filter transaction snapshots using search criteria](#).

## Learn More

- [Configure Transaction Snapshots](#)
- [Call Graphs](#)
- [Diagnostic Sessions](#)
- [Configure Data Collectors](#)
- [Configure Call Graphs](#)
- [Access the Administration Console](#)