.

Using Getter Chains

On this page:

- About Getter Chains
- Special Characters in Getter Chains
- Escape Literal Characters
- .NET Notes and Examples
- Java Notes and Examples

Related pages:

- Transaction Detection Rules

Getter chains let you access method data in various contexts of the AppDynamics configuration. For example, getter chains enable you to name business transactions based on return values.

This topic describes how to use getter chains.

About Getter Chains

You can use getter chains to:

- Create a new JMX Metric Rule and define metrics from MBean attributes. See Configure JMX Metrics from MBeans.
- Configure method invocation data collectors. See the Configuration Notes in Data Collectors.
- Define a new business transaction custom match rule that uses a POJO object instance as the mechanism to name the transaction. See POJO Entry Points.
- Configure a custom match rule for servlet entry points and name the transaction by defining methods in a getter chain. See "Split by POJO Method Call" on Split Servlet Transaction by Payload Examples.

As a best practice, you should use getter chains with simple getter methods only. Getter chains on processing-intensive methods, such as one that make numerous SQL calls, can result in degraded performance for the application and agent.

For example, the following shows a simple get method that returns a property for a class, such as `MyUser.Name`:

```
public class MyUser
{
   private String Name {get; set;}
   private String Url;
   public GetUrl() {
      return this.Url;
   }
}
```

Special Characters in Getter Chains

Use the following special characters as indicated:

- Parentheses `()` to enclose parameters
- Commas `,` to separate parameters
- Forward slashes `/` to separate type declarations from a value in a parameter
- Dots `.` to separate methods and properties in the getter chain
- Dot `.` to represent "anything" must be escaped

- Curly braces `{ }` to delineate getter chains from static elements in custom expressions on HTTPRequest objects (including in the Java Servlet Transaction Naming Configuration window and in the Split Transactions Using Request Data tab of the servlet custom match and exclude rules)

Getter chains also treat spaces at the beginning or end of a string as special characters.

Escape Literal Characters

If the getter chain should treat a special character literally, escape it using a backslash. For parentheses, you only need to escape the closing parenthesis in a string.

- The following example shows the how to escape the dot in the string parameter.

  ```
  GetAddress().GetParam(a\.b\.c\.)
  ```

  The agent executes `GetParam("a.b.c.")` on the result of `GetAddress()` and returns the value of of the parameter.

- In the following example, the first dot is part of the string method parameter, which requires an escape character. The second and third dots don't require an escape character because they are used as method separators.

  ```
  GetUser(suze\.smith).GetGroup().GetId()
  ```

- The following example shows how to escape the closing parenthesis in a search for "()" within a string.

  ```
  GetString.Find((\))
  ```

.NET Notes and Examples

The following sections apply to getter chains used in .NET Agent configurations.

Declare Parameter Types

The .NET Agent identifies parameter types as follows:

- Dictionaries, anything with property accessors, use a normal parameter set, which defaults to string.
- Arrays use single integers as parameters.

Therefore `0` means `string/0` if you have a hash, or anything that translates to `get_Item` (any kind of property). `0` means `int/0` if you have an array.

When your getter chain uses a method with parameters other than the default type, you need to declare the parameter type.

- The following example demonstrates how to declare the parameter types to resolve the overloaded `Susbstring()` method. The forward slash serves as the type separator.

  ```
  GetAddress(appdynamics, sf).Substring(int/0, int/10)
  ```

  For instance, if `GetAddress(appdynamics, sf)` returns "303 2nd St, San Francisco, CA 94107", the full getter chain expression returns "303 2nd St".

- The following example shows how to declare the parameter types for a user-defined method that takes a float parameter, a boolean parameter, and an integer parameter. The forward slash serves as the type separator.

```
GetAddress(appdynamics, sf).MyMethod(float/0\.2, boolean/true, boolean/false, int/5)
```

Access Indexed Properties and Dictionary Values

If your getter chain accesses a dictionary object, you can access the values using the following syntax:

- The following example returns the value for the key "suze.smith".

```
UserDictionary.[string/suze\.smith]
```

- The following example returns the value for the key "suze.smith" using the implied getter.

```
UserDictionary.get_Item(suze\.smith)
```

Split by Character or Regular Expression Match in .NET Getter Chains

You can split values matched by character or as a string by matching a regular expression pattern. The result of a split operation is a string or character array that you can reference in your getter chain by array index value.

The following examples illustrate how to use the character and string regular expression split operations in getter chains.

Split by Character

- The following chain splits a URL on the forward slash character. In this case the first slash acts as a type separator. The getter chain returns the fourth item in the array.

```
GetUrl().Split(char[]//).[3]
```

The agent returns "Search" when it applies the getter chain to the following URL: http://howdyworld.example.com/Search/Airfare

- In the following example, the split occurs on the forward slash character, and the result is the length of the resulting array.

```
GetUrl().Split(char[]//).Length
```

- This example illustrates a transaction splitting rule for URIs that use a semicolon delimiter. The getter chain splits on the forward slash, then splits the fourth element on the semicolon.

```
GetUri().Split(char[]//).[3].Split(char[]/;).[0]
```

The agent returns "create-user" when it applies the getter chain to the following URL: http://HowdyWorld.example.com/create-user;sessionid=BE7F31CC0235C796BF8C6DF3766A1D00?act=Add&uid=c42ab7ad-48a7-4353-bb11-0dfeabb798b5

Split by Regular Expression

For more control, you can split values by string-based pattern matching. Pattern matching is particularly useful for situations that require complex matching, such as matching content within a request body.

The following example shows a getter chain that splits the value returned by `GetAddress()` and selects the seventh element in the resultant array:

```
GetAddress().Split(string/\W+).[6]
```

Given an address such as "303 2nd St, San Francisco, CA 94107", the example splits the value by word and references the sixth word in the array, "CA" in this case.

Compose Getter Chains for HTTP Requests

The .NET Agent requires special syntax for getter chains for HTTP Requests:

- For ASP.NET WebForms, MVC, and MVC WebAPI applications, create getter chains based upon the `System.Web.HttpRequest` objects.
- For ASP.NET Core on the full framework, create getter chains based upon Microsoft.AspNetCore.Http.Internal.DefaultHttpRequest objects.

If you have both ASP.NET and ASP.NET Core on the full framework apps in the same tier, you cannot use a getter chain to apply to both because the two frameworks use different objects to handle HTTP requests.

- Use the following syntax to delineate the boundaries of the getter chain:

```
${myobject.myproperty}
```

- The following example determines the user principal:

```
${Context.User.Identity.Name}
```

Places to use this syntax include:

- HTTP Request Data Collectors
- ASP.NET Transaction Detection custom expressions

Java Notes and Examples

The following sections apply to getter chains used in Java Agent configurations.

Values Passed in a Getter Chain

The value passed in a getter chain is always a string unless cast to another type.

The following cast types are supported:

- int
- float
- bool (the primitive Boolean value)
- boolean (a boxed boolean value; i.e. an object type that wraps a boolean)
- long
- object

The following section shows examples of how to refer to the types in parameters to getter chain methods. Notice that letter case is not important for the names of the types. Type casing is performed in a case-insensitive manner.

Java Getter Chain Examples

- Getter chain with integer parameters in the substring method using the forward slash as the type separator:

```
getAddress(appdynamics, sf).substring(int/0, int/10)
```

- Getter chain with various non-string parameter types:

```
getAddress(appdynamics, sf).myMethod(float/0.2, boolean/true, boolean/false, int/5)
```

- Getter chain with forward slash escaped; escape character needed here for the string parameter:

```
getUrl().split(\/) # node slash is escaped by a backward slash
```

- Getter chain with an array element:

```
getUrl().split(\/).[4]
```

- Getter chains that return Hashmap values:

```
get(object/myvalue).substring(int/0,int/10)
get(object/ACTION)
```

- Getter chain with multiple array elements separated by commas:

```
getUrl().split(\/).[1,3]
```

- Getter chain retrieves property values, such as the length of an array:

```
getUrl().split(\.).length
```

- Getter chain using backslash to escape the dot in the string parameter;
  the call is getParam (a.b.c).

```
getAddress.getParam(a\.b\.c\.)
```

- In the following getter chain, the first dot requires an escape character because it is in a string method parameter (inside the parentheses). The second dot does not require an escape character because it is not in a method parameter (it is outside the parentheses).

```
getName(suze\.smith)getClass().getSimpleName()
```

- The following getter chain is from a transaction splitting rule on URIs that use a semicolon as a delimiter.

```
getRequestURI().split(\/).[2].split(;).[0]
```

The call gets the URI using getRequestURI() and then splits it using the escaped forward slash. From the resulting array it takes the third entry (as the split treats the first slash as a separator) and inserts what before the slash (in this case, nothing) into the first entry. Then it splits this result using the semicolon, getting the first entry of the resulting array, which in this case contains the API name. For example, given the following URI:

```
/my-webapp/xyz;jsessionid=BE7F31CC0235C796BF8C6DF3766A1D00?act=Add&uid=c42ab7ad-48a7-4353-bb11-
0dfeabb798b5
```

The getter chain splits on the API name, so the resulting split transactions are "API.abc", "API.xyz" and so on.

Tip: When using string.split(), remember that it takes a regex and you have to escape any special regex characters.

For example, if you want to split on left square bracket ([):

```
Java syntax: split("
[")
Getter chain syntax: split([)
```