This document supports the version of each product listed and supports all subsequent versions until the document is replaced by a new edition. To check for more recent editions of this document, see http://www.vmware.com/support/pubs.
You can find the most up-to-date technical documentation on the VMware Web site at:
http://www.vmware.com/support/
The VMware Web site also provides the latest product updates.
If you have comments about this documentation, submit your feedback to:
docfeedback@vmware.com
Contents

VMware vFabric AppInsight Developer’s Guide  5

1  AppInsight API Overview  7

2  Using the AppInsight API  9
   Using Endpoints to Retrieve Data  10

3  API Calls  13
   Get All Notifications on Application  13
   Get All Notifications  14
   Get the Health of All Applications  15
   Get the Health of an Application  16
   Get a Specific KPI State for an Application Over Time  16
   Add an Application  17
   Add Tier To Application  18
   Add a Component to a Tier  18
   Get Potential Components  19

4  AppInsight Plug-In Reference  21
   Server-Based Instrumentation  23
   Annotation Plug-In  23
   Apache-Http-Client3 Plug-In  24
   Apache-Http-Client4 Plug-In  24
   Blazeds Plug-In  24
   Eclipse-Persistence Plug-In  24
   Ehcache Plug-In  25
   EJB3 Plug-In  25
   Files-Tracker Plug-In  25
   Gemfire Plug-In  25
   Grails Plug-In  25
   Hibernate Plug-In  26
   JAX-RS Plug-In  26
   JDBC Plug-In  27
   JMS Plug-In  27
   JPA Plug-In  27
   JTA Plug-In  27
   JWS Plug-In  28
   LDAP Plug-In  28
   Logging Plug-In  28
   Mail Plug-In  28
   Mongodb Plug-In  28
Quartz-Scheduler Plug-In 29
Rabbitmq-Client Plug-In 29
Redis Plug-In 29
RMI Plug-In 30
Run-Exec Plug-In 30
Servlet Plug-In 30
Socket Plug-In 31
spring-batch Plug-in 31
spring-core Plug-In 31
spring-integration Plug-In 31
spring-security Plug-In 32
spring-tx Plug-In 32
spring-web Plug-In 32
spring-webflow Plug-In 33
Struts2 Plug-In 33
Tomcat Plug-In 33

Index 35
The VMware vFabric AppInsight Developer’s Guide comprises to sections. The first section describes an API that you can use to interact with the AppInsight datastore of your monitored applications, outside of the user interface. The second section describes plug-ins that you can use to gather additional metrics.

Intended Audience

This information is intended for anyone who wants to use automated processes to retrieve data from the AppInsight datastore or to perform certain topology actions. You should be familiar with REST concepts and with the JSON serialization format.

It is also intended for developers who need to incorporate plug-ins to gather metrics that are not included in AppInsight. Developers should be familiar with VMware SpringSource Spring Insight methodologies.
AppInsight API Overview

With the vFabric AppInsight API, you can retrieve data from applications that AppInsight monitors without launching the AppInsight application. You can also add objects to your application's topology.

The AppInsight API uses RESTful services. Responses use the JSON serialization format.
Using the AppInsight API

Use this information to interact with the AppInsight API.

General Requirements

Before you use the AppInsight API, note the following requirements.

- All calls must be authenticated
- You must use the GET HTTP method for retrieving data, and POST for adding data.
- The path variable is always required. It is used to focus on a single resource in the resource list.
  For example,
  
  `/applications/+APP_NAME/metrics`
  
  where `APP_NAME` is the path variable.
- All resources are defined using the plural form, for example applications, metrics, notifications, and so on.
- Query parameters may be required or optional. They are primarily used to provide a filter, or to narrow results to match criteria.
  For example,
  
  `/applications/+myAPP/metrics/health/average`
  
  where `indicator` is the query parameter.
  
  When a query parameter is optional, the default value is used if you do not specify a value.
- When traversing applications, to retrieve data for a specific application, you must indicate the application by using `+`.
  
  For example,
  
  `/applications/+myAPP/notifications`
- The URL that you use with the API cannot end with `/`.

Supported HTTP Methods

There are two HTTP methods supported by this API.
Table 2-1. Supported HTTP Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Retrieving data</td>
</tr>
<tr>
<td>POST</td>
<td>Adding data</td>
</tr>
</tbody>
</table>

Authentication

This API's authentication is based on basic HTTP or HTTPS authentication.

No session objects are created when you connect to the API. A client must send the basic HTTP headers each time that it connects to the API.

Responses

Responses to API calls differ, depending on whether the response relates to a single object or to a collection.

This is a sample response for calls that return a single object.

```json
{ "meta" : {
    "message" : "" // reserved for API messages
},
"result" : {
    // key-value fields
}
}
```

This is a sample response for calls that return a collection object.

```json
{ "meta" : {
    "message" : "" // reserved for API messages
},
"result" : [
    {
    // key-value fields for each result
}
]
}
```

Error Messages

Errors include a corresponding HTTP status code.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>HTTP Status Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad user input</td>
<td>BAD_REQUEST (400)</td>
</tr>
<tr>
<td>Unauthorized operation</td>
<td>FORBIDDEN (403)</td>
</tr>
<tr>
<td>Unrecognized error</td>
<td>SERVER_ERROR (500)</td>
</tr>
</tbody>
</table>

Using Endpoints to Retrieve Data

You use endpoints to retrieve data for a specific object from applications that AppInsight monitors.

You can append endpoints to a URL to retrieve data. You must include an item from the Data to Retrieve column for the request to be answered.
### Table 2-2. Endpoints for Retrieving Data

<table>
<thead>
<tr>
<th>Resource</th>
<th>General</th>
<th>Data to Retrieve</th>
</tr>
</thead>
<tbody>
<tr>
<td>applications</td>
<td>/</td>
<td>notifications</td>
</tr>
<tr>
<td></td>
<td></td>
<td>metrics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>average</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overtime</td>
</tr>
<tr>
<td>applications/+APP_NAME</td>
<td>/</td>
<td>metrics</td>
</tr>
<tr>
<td></td>
<td></td>
<td>average</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overtime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>notifications</td>
</tr>
<tr>
<td>infrastructure</td>
<td></td>
<td>notifications</td>
</tr>
</tbody>
</table>

The following examples are requests that use endpoints.

- https://appinsight IP:8443/am-apm-web/resources/applications/notifications
The AppInsight API includes these requests.

- **Get All Notifications on Application** on page 13
  The Get all notifications on Application query retrieves all notifications for an application during a specified time frame.

- **Get All Notifications** on page 14
  The Get all notifications query retrieves all notifications for applications during a specified time frame.

- **Get the Health of All Applications** on page 15
  The Get health of applications query retrieves the average health metrics indicator for all applications for the specified time range.

- **Get the Health of an Application** on page 16
  The Get health of application query retrieves the average health metrics indicator for an application during a specified time frame.

- **Get a Specific KPI State for an Application Over Time** on page 16
  The Get KPI state for application over time query retrieves the metrics for an indicator during a specified time frame.

- **Add an Application** on page 17
  The Add application call adds a new application to AppInsight.

- **Add Tier To Application** on page 18
  The Add Tier To Application call adds a tier to the topology of a specific application.

- **Add a Component to a Tier** on page 18
  The Add Component To Tier call adds a component to a tier in the topology of an application.

- **Get Potential Components** on page 19
  The Get Potential Components query retrieves a list of potential components that can be added to the topology of the application.

### Get All Notifications on Application

The Get all notifications on Application query retrieves all notifications for an application during a specified time frame.

You must have View permissions for the application to view the notifications.
The Get all notifications on Application query uses the following syntax.
https://appinsight_ip:8443/am-apm-web/resources/applications/notifications

**HTTP Method**

GET

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>(Required) Name of application.</td>
<td>travel</td>
</tr>
<tr>
<td>startTime</td>
<td>Sets the beginning of the time frame for fetched notifications in milliseconds. Default is 10 minutes before endTime.</td>
<td>1234098437</td>
</tr>
<tr>
<td>endTime</td>
<td>Sets the end of the time frame for fetched notifications in milliseconds. Default is current time.</td>
<td>1235098344</td>
</tr>
</tbody>
</table>

**Response**

```
{ "meta" :{}, "result" : [ {"target": {"title":"travel", "type":"application"},
  "source":"appinsight",
  "type":"alert",
  "message":"At 7:40 AM, the Performance KPI deteriorated to Warning state",
  "eventTime", "1234098500"},
  {...}
  ]}
```

**Get All Notifications**

The Get all notifications query retrieves all notifications for applications during a specified time frame.
You can see notifications only for applications for which you have View permissions.
The notifications are at application level and do not include infrastructure or middleware notifications.
Notifications are sorted by their event time. You can retrieve the name of the application to which a notification refers from the notification message.
The Get all notifications query uses the following syntax.
/resources/applications/notifications

**HTTP Method**

GET

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>startTime</td>
<td>Sets the beginning of the time frame for fetched notifications in milliseconds. Default is 10 minutes before endTime.</td>
<td>1234098437</td>
</tr>
<tr>
<td>endTime</td>
<td>Sets the end of the time frame for fetched notifications in milliseconds. Default is current time.</td>
<td>1235098344</td>
</tr>
</tbody>
</table>
Response

```json
{ "meta":{}, "result": [ { "target": { "title": "travel", "dataObjectType": "application"}, "source": "appinsight", "type": "alert", "message": "At 7:40 AM, the Performance KPI deteriorated to Warning state", "eventTime": "1234098500" }, {...} ]
}
```

Get the Health of All Applications

The `Get health of applications` query retrieves the average health metrics indicator for all applications for the specified time range.

You can see data only for applications for which you have View permissions.

The `Get health of applications` query uses the following syntax.

```
https://appinsight_ip:8443/am-apm-web/resources/applications/metrics/METRIC/average
```

**HTTP Method**

GET

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>METRIC</td>
<td>(Required) The metric that is to be fetched for all applications.</td>
<td>health</td>
</tr>
<tr>
<td>startTime</td>
<td>Sets the beginning of the time frame for fetched notifications in milliseconds. Default is 10 minutes before endTime.</td>
<td>1234098437</td>
</tr>
<tr>
<td>endTime</td>
<td>Sets the end of the time frame for fetched notifications in milliseconds. Default is current time.</td>
<td>1235098344</td>
</tr>
</tbody>
</table>

**Response**

```json
{ "meta":{}, "result": [ { "metric": "HEALTH", "state": "WARNING", "element": { "title": "travel", "description": "description", "dataObjectType": "APPLICATION" } }, { "metric": "HEALTH", "state": "EXCELLENT", "element": { "title": "petclinic", "description": "description", "dataObjectType": "APPLICATION" } } ]
}
```
Get the Health of an Application

The **Get health of application** query retrieves the average health metrics indicator for an application during a specified time frame.

You must have View permissions for the application to view the data.

The **Get health of application** query uses the following syntax.

```
https://appinsight_ip:8443/am-apm-web/resources/applications/+APP_NAME/metrics/INDICATOR/average
```

**HTTP Method**

GET

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>(Required) Name of application.</td>
<td>travel</td>
</tr>
<tr>
<td>INDICATOR</td>
<td>(Required) The indicator metric that is to be fetched for the specified APP_NAME.</td>
<td>health</td>
</tr>
<tr>
<td>startTime</td>
<td>Sets the beginning of the time frame for fetched notifications in milliseconds. Default is 10 minutes before endTime.</td>
<td>1234098437</td>
</tr>
<tr>
<td>endTime</td>
<td>Sets the end of the time frame for fetched notifications in milliseconds. Default is current time.</td>
<td>1235098344</td>
</tr>
</tbody>
</table>

**Response**

```
{
  "meta":{},
  "result": [
    {
      "metric": "HEALTH",
      "state": "WARNING",
      "element": {
        "title": "travel",
        "description": "description",
        "dataObjectType": "APPLICATION"
      }
    }
  ]
}
```

Get a Specific KPI State for an Application Over Time

The **Get KPI state for application over time** query retrieves the metrics for an indicator during a specified time frame.

You must have View permissions for the application to see the state.

The **Get KPI state for application over time** query uses the following syntax.

```
https://appinsight_ip:8443/am-apm-web/resources/applications/+APP_NAME/metrics/INDEX/average
```
HTTP Method

GET

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>(Required) Name of application.</td>
<td>travel</td>
</tr>
<tr>
<td>METRIC</td>
<td>(Required) The metric that is to be fetched for the specified APP_NAME.</td>
<td>health</td>
</tr>
<tr>
<td>startTime</td>
<td>Sets the beginning of the time frame for fetched notifications in milliseconds. Default is 10 minutes before endTime.</td>
<td>1234098437</td>
</tr>
<tr>
<td>endTime</td>
<td>Sets the end of the time frame for fetched notifications in milliseconds. Default is current time.</td>
<td>1235098344</td>
</tr>
</tbody>
</table>

Response

```
{ "meta":{}, "result": [ {
   "Health KPI": "EXCELLENT",
   "occurred at": "1324566000000"
},
   {
   "Health KPI": "CRITICAL",
   "occurred at": "1324566120000"
   },
   {...}
]}
```

Add an Application

The Add application call adds a new application to AppInsight.

You must have the Administrator role to use this request.

The application name must be unique.

The Add application call uses the following syntax.

https://appinsight_ip:8443/am-apm-web/resources/applications/+APP_NAME

HTTP Method

POST

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>(Required) Name of application.</td>
<td>travel</td>
</tr>
<tr>
<td>description</td>
<td>Description of the application.</td>
<td>online travel agency application</td>
</tr>
</tbody>
</table>
Add Tier To Application

The Add Tier To Application call adds a tier to the topology of a specific application.

You must have Edit permissions to use this request.

The Add Tier To Application call uses the following syntax.

https://appinsight_ip:8443/am-apm-web/resources/applications/+APP_NAME/metrics/METRIC/overtime

HTTP Method

POST

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>Name of application (required.)</td>
<td>travel</td>
</tr>
<tr>
<td>TIER_NAME</td>
<td>Name of tier (required.)</td>
<td>load balancer</td>
</tr>
<tr>
<td>position</td>
<td>Position of the new tier. Possible values are left for left-most tier, and right for right-most tier. Default is left.</td>
<td>left</td>
</tr>
</tbody>
</table>

Response

{
  "meta":{},
  "result": [
    {"title":"load balancer","dataObjectType":"TIER"}
  ]
}

Add a Component to a Tier

The Add Component To Tier call adds a component to a tier in the topology of an application.

You must have Edit permissions to use this request.

The Add Component To Tier call uses the following syntax.

https://appinsight_ip:8443/am-apm-web/resources/applications/+APP_NAME/tiers/+TIER_NAME/components/+COMPONENT_NAME

HTTP Method

POST
## Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>(Required) Name of application.</td>
<td>travel</td>
</tr>
<tr>
<td>TIER_NAME</td>
<td>(Required) Name of tier.</td>
<td>load balancer</td>
</tr>
<tr>
<td>COMPONENT_NAME</td>
<td>(Required) Name of component.</td>
<td>travel</td>
</tr>
<tr>
<td>type</td>
<td>(Required) Type of component. Possible values are APPSPEED_ATOM, BCI_COMPONENT, and BCI_DB_COMPONENT.</td>
<td>BCI_COMPONENT</td>
</tr>
<tr>
<td>ip</td>
<td>(Required) IP of component.</td>
<td>172.16.1.12</td>
</tr>
<tr>
<td>port</td>
<td>(Required) Port number of component.</td>
<td>80</td>
</tr>
<tr>
<td>protocol</td>
<td>(Required for network components) Type of protocol. Possible values are HTTP, MSSQL, MYSQL, ORACLE, and HSQLDB.</td>
<td>HTTP</td>
</tr>
<tr>
<td>agentId</td>
<td>(Required if adding a BCI component) ID of the BCI agent.</td>
<td>169331274</td>
</tr>
</tbody>
</table>

## Response

```json

{ 
  "meta":{}, 
  "result": [
    
  ]
}
```

## Get Potential Components

The Get Potential Components query retrieves a list of potential components that can be added to the topology of the application.

You must have the Administrator role to see the potential components.

The Get Potential Components query returns the following query.

```
https://appinsight_ip:8443/am-apm-web/resources/applications/+APP_NAME
```

## HTTP Method

GET

## Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>(Required) Name of application.</td>
<td>travel</td>
</tr>
</tbody>
</table>

## Response

```json

{ 
  "meta":{}, 
  "result": [
    
      
    ]
}
```
"port":80,"ip":"10.23.202.74",
"componentProtocol":null},...

]}

}
Most metrics are gathered using plug-ins to AppInsight. You can develop your own plug-ins to gather additional metrics.

AppInsight uses the plug-ins that are provided in VMware SpringSource Spring Insight to gather metrics. You can use Spring Insight plug-ins to gather metrics that are not included in AppInsight.

Spring Insight captures application events known as traces. A trace represents a thread of execution. It is usually started by an HTTP request but can also be started by a background job. A trace contains operations. Each operation represents a significant point in the execution of the trace, for example, a JDBC query or transaction commit.

Using this data, Spring Insight calculates summary information to lead you to the specifics of why your application may not be performing optimally.

Spring Insight uses AspectJ to intercept operations in target web applications. Target web applications are loaded with a special classloader that dynamically instruments web applications at runtime. Spring Insight uses algorithms to keep the memory footprint low.

Developers who are creating their own plug-ins should be familiar with Spring Insight methodology.

For information about how to integrate Spring Insight plug-ins to AppInsight, see http://pubs.vmware.com/vfabric52/topic/com.vmware.vfabric.tc-server.2.8/devedition/about.html.

For a plug-in development tutorial, see http://pubs.vmware.com/vfabric52/topic/com.vmware.vfabric.tc-server.2.8/devedition/tutorial-plugin.html.

**Understanding Spring Insight Plug-Ins in the AppInsight Context**

When you are working with the Spring Insight plug-ins to customize your AppInsight metrics collection, you might encounter terminology that is not used in AppInsight.

<table>
<thead>
<tr>
<th>Spring Insight Term</th>
<th>AppInsight Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endpoint</td>
<td>Transaction element</td>
</tr>
<tr>
<td>Applications</td>
<td>Application components</td>
</tr>
</tbody>
</table>

This chapter includes the following topics:

- “Server-Based Instrumentation,” on page 23
- “Annotation Plug-In,” on page 23
- “Apache-Http-Client3 Plug-In,” on page 24
- “Apache-Http-Client4 Plug-In,” on page 24
- “Blazeds Plug-In,” on page 24
- “Eclipse-Persistence Plug-In,” on page 24
- “Ehcache Plug-In,” on page 25
- “EJB3 Plug-In,” on page 25
- “Files-Tracker Plug-In,” on page 25
- “Gemfire Plug-In,” on page 25
- “Grails Plug-In,” on page 25
- “Hibernate Plug-In,” on page 26
- “JAX-RS Plug-In,” on page 26
- “JDBC Plug-In,” on page 27
- “JMS Plug-In,” on page 27
- “JPA Plug-In,” on page 27
- “JTA Plug-In,” on page 27
- “JWS Plug-In,” on page 28
- “LDAP Plug-In,” on page 28
- “Logging Plug-In,” on page 28
- “Mail Plug-In,” on page 28
- “Mongodb Plug-In,” on page 28
- “Quartz-Scheduler Plug-In,” on page 29
- “Rabbitmq-Client Plug-In,” on page 29
- “Redis Plug-In,” on page 29
- “RMI Plug-In,” on page 30
- “Run-Exec Plug-In,” on page 30
- “Servlet Plug-In,” on page 30
- “Socket Plug-In,” on page 31
- “spring-batch Plug-in,” on page 31
- “spring-core Plug-In,” on page 31
- “spring-integration Plug-In,” on page 31
- “spring-security Plug-In,” on page 32
- “spring-tx Plug-In,” on page 32
- “spring-web Plug-In,” on page 32
- “spring-webflow Plug-In,” on page 33
- “Struts2 Plug-In,” on page 33
- “Tomcat Plug-In,” on page 33
Server-Based Instrumentation

The following instrumentation is provided by default to the tc Runtime. It is not provided as a plug-in.

**AbstractHttpRequestOperationSupport**

This instrumentation creates an HTTP operation for every HTTP request to a web application.

Full request and response headers are collected as provided by Tomcat and the Servlet API. Request parameters are provided if available. User parsing of the request body prevents parameters from being available.

Request headers are collected before the application is involved, and response headers are collected after the application has returned.

**ApplicationLifecycleCollectionSupport**

This instrumentation creates traces for application life cycle events such as start and stop. Not all web applications have a start event when the server is first starting, if the agent has not fully initialized.

Annotation Plug-In

Annotation plug-ins make it easy for end users to define custom operation frames and end points without the need to create an additional plug-in.

Operations of the Annotation Plug-In

Because end user code modification is required to use @Insight* annotations, they are an option for users who cannot or do not want to write aspects.

For more information about using @Insight* annotations, see [http://pubs.vmware.com/vfabric52/topic/com.vmware.vfabric.tc-server.2.8/devedition/using-insight-annotations.html](http://pubs.vmware.com/vfabric52/topic/com.vmware.vfabric.tc-server.2.8/devedition/using-insight-annotations.html).

**Table 4-2. annotation Plug-In Operations**

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@InsightOperation</td>
<td>Any methods annotated with @InsightOperation create an operation, including basic source code location, method parameters, and the return value. The operation appears in the AppInsight Summary view.</td>
</tr>
<tr>
<td>@InsightEndPoint</td>
<td>The @InsightEndPoint collection point is similar to @InsightOperation. It also indicates that the operation should be treated as an endpoint. During an update using this operation, a transaction element is created in AppInsight.</td>
</tr>
<tr>
<td>@InsightObscure</td>
<td>The @InsightObscure collection point annotates a value that must be obscured from the Samples transmission or storage.</td>
</tr>
<tr>
<td>@InsightSensitive</td>
<td>The @InsightSensitive marks a method that processes sensitive data.</td>
</tr>
</tbody>
</table>

Endpoint Analysis of the annotation Plug-In

The Annotation plug-in end points are described here.

**Analyzer**

AnnotatedMethodEndPointAnalyzer

**Core Operation**

AnnotatedMethodOperation
Score
High by default, customizable in annotation.

Summary
Finds operations created from `@InsightEndPoint`, thus creating the end point.

**Apache-Http-Client3 Plug-In**

The Apache-Http-Client3 plug-in provides trace support for calls to external HTTP resources using the Apache HTTP client version 3.

The operation of the Apache-Http-Client3 plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>HttpClient.executeMethod(...)</td>
<td>Executes HTTP calls.</td>
</tr>
</tbody>
</table>

**Apache-Http-Client4 Plug-In**

The Apache-Http-Client4 plug-in provides trace support for calls to external HTTP resources using the Apache HTTP client version 4.

The operation of the Apache-Http-Client4 plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>HttpClient.execute(...)</td>
<td>Executes HTTP calls.</td>
</tr>
</tbody>
</table>

**Blazeds Plug-In**

The Blazeds plug-in provides trace support for calls to BlazeDS. It supports the message broker APIs, and command and message based communication.

The operations of the Blazeds plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageBroker.route*(..)</td>
<td>Used for message broker operations.</td>
</tr>
<tr>
<td>Service.serviceMessage(..)</td>
<td>Used for remote object and message-based operations.</td>
</tr>
<tr>
<td>Service.serviceCommand(..)</td>
<td>Used for command-based operations.</td>
</tr>
</tbody>
</table>

**Endpoint Analysis of BlazeDS Plug-In**

The end points of the Blazeds plug-in are described here.

<table>
<thead>
<tr>
<th>Analyzer</th>
<th>BlazeDSEndPointAnalyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>Varies.</td>
</tr>
<tr>
<td>Summary</td>
<td>Creates an end point for HTTP requests handled by BlazeDS.</td>
</tr>
</tbody>
</table>

**Eclipse-Persistence Plug-In**

The Eclipse-Persistence plug-in provides trace support for calls to EclipseLink.

The operations of the Eclipse-Persistence plug-in are described in this table.
Ehcache Plug-In

The Ehcache plug-in provides trace support for calls to Ehcache.
The operations of the Ehcache plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get</td>
<td>Gets a value from Ehcache.</td>
</tr>
<tr>
<td>Put</td>
<td>Places a value in Ehcache.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes a value from Ehcache.</td>
</tr>
<tr>
<td>Replace</td>
<td>Replaces a value in Ehcache.</td>
</tr>
</tbody>
</table>

EJB3 Plug-In

The EJB3 plug-in provides trace support for calls to EJB3 beans.
The operations of the EJB3 plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Stateful</td>
<td>Execution of stateful beans.</td>
</tr>
<tr>
<td>@Stateless</td>
<td>Execution of stateful beans.</td>
</tr>
</tbody>
</table>

Files-Tracker Plug-In

The Files-Tracker plug-in provides trace support for file open and close operations.
The operation of the Files-Tracker plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>Opens a file.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes a file.</td>
</tr>
</tbody>
</table>

Gemfire Plug-In

The Gemfire plug-in provides trace support for calls to GemFire.
The operations of the Gemfire plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>Writes and reads values using the GemFire Region APIs.</td>
</tr>
<tr>
<td>Query</td>
<td>Queries GemFire using the Query APIs.</td>
</tr>
</tbody>
</table>

Grails Plug-In

The Grails plug-in provides Grails-specific support for controller methods.
The operation of the Grails plug-in is described in this table.
### Endpoint Analysis of Grails Plug-In

The end points of the Grails plug-in are here.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grails Controller Method</td>
<td>The method on a Grails Controller to which the request is mapped.</td>
</tr>
</tbody>
</table>

#### Hibernate Plug-In

The Hibernate plug-in provides trace support for persistent operations such as get, save, update, and delete.

A Hibernate session represents a single transaction, which may include several database transactions.

The Hibernate plug-in collects the following data:
- The method name, such as flush, save, update, delete, and so on.
- The entity count, which is the number of entity instances associated with the Hibernate session
- The collection count, which is the number of collection instances associated with the session

The operation of the Hibernate plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hibernate.Session</td>
<td>CRUD operation on persistent entities.</td>
</tr>
</tbody>
</table>

#### JAX-RS Plug-In

The JAX-RS plug-in provides trace support for JAX_RS.

The operations of the JAX-RS plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@DELETE</td>
<td>Collect DELETE operations</td>
</tr>
<tr>
<td>@GET</td>
<td>Collect GET operations</td>
</tr>
<tr>
<td>@HEAD</td>
<td>Collect HEAD operations</td>
</tr>
<tr>
<td>@POST</td>
<td>Collect POST operations</td>
</tr>
<tr>
<td>@PUT</td>
<td>Collect PUT operations.</td>
</tr>
</tbody>
</table>

### Endpoint Analysis of JAX-RS Plug-In

The end points of the JAX-RS plug-in are here.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JaxrsEndPointAnalyzer</td>
<td>JAX-RS plug-in provides trace support for JAX_RS.</td>
</tr>
<tr>
<td>Score</td>
<td>Varies.</td>
</tr>
<tr>
<td>Summary</td>
<td>Creates an end point for HTTP requests handled by JAX-RS.</td>
</tr>
</tbody>
</table>
JDBC Plug-In

The JDBC plug-in provides low level trace support for raw SQL queries.
The operations of the JDBC plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.sql.Statement</td>
<td>Operation with the raw SQL statement.</td>
</tr>
<tr>
<td>java.sql.PreparedStatement</td>
<td>Operation with the raw SQL statement and parameters.</td>
</tr>
</tbody>
</table>

JMS Plug-In

The JMS plug-in provides JMS tracing support.
The operations of the JMS plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageConsumer</td>
<td>Traces the javax.jms.MessageConsumer APIs.</td>
</tr>
<tr>
<td>MessageListener</td>
<td>Traces the javax.jms.MessageListener APIs.</td>
</tr>
<tr>
<td>MessageProducer</td>
<td>Traces the javax.jms.MessageProducer APIs.</td>
</tr>
</tbody>
</table>

Endpoint Analysis of JMS Plug-In

The JMS plug-in end points are described here.

<table>
<thead>
<tr>
<th>Analyzer</th>
<th>JMSConsumerEndPointAnalyzer and JMSMessageListenerEndPointAnalyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score</td>
<td>Varies.</td>
</tr>
<tr>
<td>Summary</td>
<td>Creates an end point for messages that are received on a queue or topic.</td>
</tr>
</tbody>
</table>

JPA Plug-In

The JPA plug-in provides trace support for the Java Persistence API.
The operations of the JPA plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.persistence.EntityManager</td>
<td>Traces entity and lifecycle operations.</td>
</tr>
<tr>
<td>javax.persistence.EntityTransaction</td>
<td>Traces transaction control on resource-level entity managers.</td>
</tr>
</tbody>
</table>

JTA Plug-In

The JTA plug-in provides trace support for the Java Transaction API.
The operations of the JTA plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.transaction.Transaction</td>
<td>Traces basic transaction operations.</td>
</tr>
<tr>
<td>javax.transaction.TransactionManager</td>
<td>Traces application server level transaction operations.</td>
</tr>
<tr>
<td>javax.transaction.UserTransaction</td>
<td>Traces application level transaction operations.</td>
</tr>
</tbody>
</table>
JWS Plug-In

The JWS plug-in provides trace support for JWS.
The operations of the JWS plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@WebService</td>
<td>Traces the <code>javax.jws.WebService</code> APIs.</td>
</tr>
</tbody>
</table>

Endpoint Analysis of JWS Plug-In

The JWS plug-in end points are described here.

<table>
<thead>
<tr>
<th>Analyzer</th>
<th>Score</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JwsEndPointAnalyzer</td>
<td></td>
<td>Creates an end point for HTTP requests handled by JAX-RS.</td>
</tr>
</tbody>
</table>

LDAP Plug-In

The LDAP plug-in provides trace support for LDAP lookup calls.
The operation of the LDAP plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>javax.naming.directory.DirContext</code></td>
<td>Traces LDAP lookup calls.</td>
</tr>
</tbody>
</table>

Logging Plug-In

The Logging plug-in provides support for logging frameworks. Log4J, SLF4J and Commons logging is supported. This plug-in creates traces when an error or fatal message is logged.
The operations of the Logging plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Log.error</code> and <code>Log.fatal</code></td>
<td>Logs error traces using Commons logging.</td>
</tr>
<tr>
<td><code>Category.error</code> and <code>Category.fatal</code></td>
<td>Logs error traces using Log4J.</td>
</tr>
<tr>
<td><code>Logger.error</code></td>
<td>Logs error traces using SLF4J.</td>
</tr>
</tbody>
</table>

Mail Plug-In

The Mail plug-in provides trace support for mail sent using Java mail.
The operation of the Mail plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>javax.mail.Transport.sendMessage(..)</code></td>
<td>Sends an email message.</td>
</tr>
</tbody>
</table>

Mongodb Plug-In

The Mongodb plug-in provides trace support for MongoDB.
The operations of the Mongodb plug-in are described in this table.
Quartz-Scheduler Plug-In

The Quartz-Scheduler plug-in provides trace support for Quartz scheduler queries. The operations of the Quartz-Scheduler plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job.execute</td>
<td>Quartz job execution.</td>
</tr>
</tbody>
</table>

Endpoint Analysis of Quartz-Scheduler Plug-In

The Quartz-Scheduler plug-in end points are described here.

Analyzer: QuartzSchedulerEndPointAnalyzer
Score: Low.
Summary: Creates an end point for Quartz-triggered jobs.

Rabbitmq-Client Plug-In

The Rabbitmq-Client plug-in provides support for tracing RabbitMQ queries. The operations of the Rabbitmq-Client plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
</table>

Endpoint Analysis of Rabbitmq-Client Plug-In

The rabbitmq-client plug-in end points are described here.

Analyzer: AbstractRabbitMQResourceAnalyzer
Score: Low.
Summary: Creates an end point for messages that are received on an exchange or routing key.

Redis Plug-In

The Redis plug-in provides support for tracing Redis queries. The operations of the Redis plug-in are described in this table.
Collection Point | Summary
--- | ---
Jedis | Collects operations executed using the Jedis APIs.
AbstractRedisCollection | Collects operations executed using the Spring Data abstraction for Redis.

**RMI Plug-In**

The RMI plug-in provides trace support for Java API for Remote Method Invocation (RMI.)

The operation of the RMI plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
</table>
| java.rmi.registry.Registry | Traces RMI registry operations.

**Run-Exec Plug-In**

The Run-Exec plug-in provides trace support for tracking task execution and scheduling.

The operations of the Run-Exec plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.concurrent.Executor</td>
<td>Traces concurrent task execute operations.</td>
</tr>
<tr>
<td>java.util.concurrent.ExecutorService</td>
<td>Traces concurrent task submission operations.</td>
</tr>
<tr>
<td>java.util.concurrent.ScheduledExecutorService</td>
<td>Traces concurrent task scheduling operations.</td>
</tr>
<tr>
<td>org.springframework.core.task.AsyncTaskExecutor</td>
<td>Traces concurrent task execution and submission operations.</td>
</tr>
<tr>
<td>java.lang.Thread</td>
<td>Traces thread start operations.</td>
</tr>
<tr>
<td>java.util.Timer</td>
<td>Traces task scheduling operations.</td>
</tr>
</tbody>
</table>

**Servlet Plug-In**

The Servlet plug-in creates end points for servlets and application lifecycle events such as start and stop. Support for servlet Listeners and Filters operation collection. For tc Runtime, the collection is made by the com.springsource.insight.collection.tcserver packages.

The operations of the Servlet plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.servlet.Filter</td>
<td>Operation with the filter name and initialization parameters.</td>
</tr>
<tr>
<td>javax.servlet.ServletContextListener</td>
<td>Operations for context initialization and destruction events with context parameters.</td>
</tr>
</tbody>
</table>

**Endpoint Analysis of Servlet Plug-In**

The end points of the servlet plug-in are described here.

**Analyzer** | ServletEndPointAnalyzer
**Core Operation** | HttpOperation
**Score** | Low.
**Summary** | Creates an end point for HTTP requests, grouped by the matching servlet, that are otherwise not accounted for by an end point.
Socket Plug-In

The Socket plug-in provides trace support for opening and closing of sockets.

The operations of the Socket plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>HttpURLConnection</td>
<td>Creates a new connection.</td>
</tr>
<tr>
<td>ServerSocketChannel, ServerSocket</td>
<td>Accepts a connection.</td>
</tr>
<tr>
<td>SocketChannel</td>
<td>Creates a new connection or opens a connection.</td>
</tr>
<tr>
<td>Socket</td>
<td>Creates a connection.</td>
</tr>
</tbody>
</table>

spring-batch Plug-in

The spring-batch plug-in provides trace support for calls to the spring-batch framework.

The operations of the spring-batch plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.springframework.batch.core.Job</td>
<td>Traces job execution operations.</td>
</tr>
<tr>
<td>org.springframework.batch.core.Step</td>
<td>Traces step execution operations.</td>
</tr>
<tr>
<td>org.springframework.batch.core.job.flow.Flow</td>
<td>Traces flow start and resume operations.</td>
</tr>
<tr>
<td>org.springframework.batch.core.job.flow.FlowExecutor</td>
<td>Traces flow execution operations.</td>
</tr>
</tbody>
</table>

spring-core Plug-In

The spring-core plug-in provides basic support for calls to Spring-managed beans containing user business logic. Only calls to objects that are annotated in the @Service, or @Repository stereotype annotations are supported. Even if component scanning is not enabled, you can use these annotations so that the agent picks up the operations.

The operations of the spring-core plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Repository</td>
<td>Creates an operation for calls into classes annotated with @Repository.</td>
</tr>
<tr>
<td>@Service</td>
<td>Creates an operation for calls into classes annotated with @Service.</td>
</tr>
</tbody>
</table>

spring-integration Plug-In

The spring-integration plug-in provides Spring integration support.

The operations of the spring-integration plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageChannel</td>
<td>Sends messages using the MessageChannel APIs.</td>
</tr>
<tr>
<td>MessageHandler</td>
<td>Handles messages using the MessageChannel APIs.</td>
</tr>
<tr>
<td>Transformer</td>
<td>Transforms messages using the Transformer APIs.</td>
</tr>
</tbody>
</table>
Endpoint Analysis of spring-integration Plug-In

The end points of the spring-integration plug-in are described here.

**Analyzer** IntegrationEndPointAnalyzer

**Score** Low.

**Summary** Creates an end point for operations that were triggered in Spring Integration.

spring-security Plug-In

The spring-security plug-in provides trace support for calls to Spring Security.

The operations of the spring-security plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticationManager</td>
<td>Calls to the authenticate operation.</td>
</tr>
<tr>
<td>AuthenticationProvider</td>
<td>Calls to the authenticate operation.</td>
</tr>
<tr>
<td>UserDetailsService</td>
<td>Calls to the loadUserByUsername, createUser, updateUser, deleteUser, changePassword and userExists operations.</td>
</tr>
</tbody>
</table>

spring-tx Plug-In

The spring-tx plug-in detects the creation, commit, and rollback of a transaction using the Spring PlatformTransactionManager. Transaction boundaries defined with @Transactional or with standard Spring config are detected.

The operation of the spring-tx plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlatformTransactionManager</td>
<td>Defines the boundaries of a transaction.</td>
</tr>
<tr>
<td></td>
<td>Attributes such as the propagation,</td>
</tr>
<tr>
<td></td>
<td>isolation, timeout and readonly states are</td>
</tr>
<tr>
<td></td>
<td>collected.</td>
</tr>
</tbody>
</table>

spring-web Plug-In

The spring-web plug-in provides full support for events in the Spring MVC request life cycle. Both annotated @Controller and the legacy Controller interface class hierarchy types of MVC are supported.

The operations of the spring-web plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@InitBinder</td>
<td>Data binder configuration including target name and type, allowed,</td>
</tr>
<tr>
<td></td>
<td>required, and disallowed fields.</td>
</tr>
<tr>
<td>@ModelAttribute</td>
<td>Methods returning objects that are added directly to the model,</td>
</tr>
<tr>
<td></td>
<td>including the attribute name and value.</td>
</tr>
<tr>
<td>@RequestMapping</td>
<td>Spring MVC request handlers.</td>
</tr>
<tr>
<td>Validator</td>
<td>Validation logic.</td>
</tr>
<tr>
<td>View</td>
<td>View rendering including view name and content type.</td>
</tr>
<tr>
<td>ViewResolver</td>
<td>View resolution with the view name requested and the matching View</td>
</tr>
<tr>
<td></td>
<td>object returned.</td>
</tr>
<tr>
<td>Controller</td>
<td>Legacy Spring MVC Controller interface.</td>
</tr>
<tr>
<td>DispatcherServlet</td>
<td>Handling of requests by Spring's DispatcherServlet.</td>
</tr>
</tbody>
</table>
Endpoint Analysis of spring-web Plug-In

The end points of the spring-web plug-in are described here.

<table>
<thead>
<tr>
<th>Analyzer</th>
<th>ControllerEndPointAnalyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Operation</td>
<td>ControllerMethodOperation</td>
</tr>
<tr>
<td>Score</td>
<td>Varies.</td>
</tr>
<tr>
<td>Summary</td>
<td>Creates an end point for HTTP requests handled by Spring MVC.</td>
</tr>
</tbody>
</table>

spring-webflow Plug-In

The spring-webflow plug-in provides trace support for calls to the Spring WebFlow classes.

The operations of the spring-webflow plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionExecutor</td>
<td>Traces ActionExecutor helper class execution operations.</td>
</tr>
<tr>
<td>FlowExecutionImpl</td>
<td>Traces webflow execution start and state operations.</td>
</tr>
<tr>
<td>Transition</td>
<td>Traces webflow state transition operations.</td>
</tr>
</tbody>
</table>

Struts2 Plug-In

The Struts2 plug-in provides trace support for calls to the Struts2 framework.

The operations of the Struts2 plug-in are described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Action</td>
<td>Creates an operation for calls to classes that are annotated with @Action.</td>
</tr>
<tr>
<td>ActionSupport</td>
<td>Traces Struts2 operations.</td>
</tr>
<tr>
<td>Interceptor</td>
<td>Traces Struts2 custom interceptors.</td>
</tr>
<tr>
<td>ServletDispatcherResult</td>
<td>Traces Struts2 overall action results.</td>
</tr>
<tr>
<td>ActionProxy</td>
<td>Traces Struts2 flow execution starts.</td>
</tr>
</tbody>
</table>

Tomcat Plug-In

The Tomcat plug-in provides server instrumentation for Tomcat and vFabric tc Server internal processes that are not exposed by the public Servlet API.

The operation of the Tomcat plug-in is described in this table.

<table>
<thead>
<tr>
<th>Collection Point</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP Compiler</td>
<td>Operation showing the time taken in JSP compilation (typically the first request to a JSP) and the compiler implementation.</td>
</tr>
</tbody>
</table>
Index

A
Apache-Http-Client3 plug-in 24
Apache-Http-Client4 plug-in 24
api
  Add application 17
  Add Component To Tier 18
  Add Tier To Application 18
  errors 9
  Get all notifications 14
  Get all notifications on Application 13
  Get health of application 16
  Get health of applications 15
  Get KPI state for application over time 16
  Get Potential Components 19
  list of calls 13
  overview 7
  responses 9
  supported HTTP methods 9
  using 9
  using endpoints 10

B
Blazeds plug-in 24

E
Eclipse-Persistence plug-in 24
Ehcache plug-in 25
EJB3 plug-in 25
endpoints, using to retrieve data 10

F
Files-Tracker plug-in 25

G
Gemfire plug-in 25
Grails plug-in 25

H
Hibernate plug-in 26

I
instrumentation, server based 23

J
JAX-RS plug-in 26
JDBC plug-in 27
JMS plug-in 27
JPA plug-in 27
JTA plug-in 27
JWS plug-in 28

L
LDAP plug-in 28
Logging plug-in 28

M
Mail plug-in 28
Mongodb plug-in 28

P
plug-ins
  Annotations 23
  Apache-Http-Client3 24
  Apache-Http-Client4 24
  Blazeds 24
  custom end points 23
  custom operation frames 23
  Eclipse-Persistence 24
  Ehcache 25
  EJB3 25
  Files-Tracker 25
  Gemfire 25
  Grails 25
  Hibernate 26
  JAX-RS 26
  JDBC 27
  JMS 27
  JPA 27
  JTA 27
  JWS 28
  LDAP 28
  Logging 28
  Mail 28
  Mongodb 28
  Quartz-Scheduler 29
  Rabbitmq-Client 29
  Redis 29
  reference 21
  RMI 30
  Run-Exec 30
Servlet 30
socket 31
spring-batch 31
spring-core 31
spring-integration 31
spring-security 32
spring-tx 32
spring-web 32
spring-webflow 33
Struts2 33
Tomcat 33

Q
Quartz-Scheduler plug-in 29

R
Rabbitmq-Client plug-in 29
Redis plug-in 29
RMI plug-in 30
Run-Exec plug-in 30

S
server based instrumentation 23
Servlet plug-in 30
Socket plug-in 31
spring-batch plug-in 31
spring-core plug-in 31
spring-integration plug-in 31
spring-security plug-in 32
spring-tx plug-in 32
spring-web plug-in 32
spring-webflow plug-in 33
Struts2 plug-in 33

T
Tomcat plug-in 33